Deep Fisher Faces

Harald Hanselmann
hanselmann@cs.rwth-aachen.de
Shen Yan
shen.yan@rwth-aachen.de
Hermann Ney
ney@cs.rwth-aachen.de

Human Language Technology and
Pattern Recognition Group
RWTH Aachen University
Aachen, Germany

Abstract

Most current state-of-the-art methods for unconstrained face recognition use deep convolutional neural networks. Recently, it has been proposed to augment the typically used softmax cross-entropy loss by adding a *center loss* trying to minimize the distance between the face images and their class centers. In this work we further extend the center (intra-class) loss with an inter-class loss reminiscent of the popular early face recognition approach *Fisherfaces*. To this end we add a term that directly optimizes the distances of the class centers appearing in a batch in dependence of the input images. We evaluate the new loss on two popular databases for unconstrained face recognition, the Labeled Faces in the Wild and the Youtube Faces database. In both cases the new loss achieves competitive results.

1 Introduction

The field of face recognition has seen a significant boost in performance since the emergence of deep learning techniques in the area [10, 13, 15, 17]. Today, many state-of-the-art approaches use Convolutional Neural Networks (CNNs) in one way or another. In the context of the specific task of face verification, where two face images are given and it has to be determined whether they belong to the same identity or not, a popular approach is to train a deep model that maps the face images into an embedding space [5, 12, 13, 16, 17]. This space is then used to compute the similarity of the given images. Specifically, in [17] the authors propose the center loss that aims to minimize the distances between the image features (network output) and their class centers in the embedding space. The model is trained using backpropagation and a global center is maintained for each class by updating the centers after each iteration using only the features in the current batch of the training procedure.

In this paper we aim to extend the intra-class loss provided by the center loss with an inter-class loss that simultaneously maximizes the distances between the classes. To make this possible, we first create a direct functional dependence between the features of a batch and their centers within the same iteration (Section 3.1). This step is necessary to formulate an inter-class loss based on only the global centers, while still obtaining a gradient that directly influences all network parameters. Next we add the inter-class loss (Section 3.2) and evaluate the proposed method on two popular benchmark databases for unconstrained face recognition, the Labeled Faces in the Wild (LFW) [7] and the Youtube Faces (YTF) [18] databases (Section 4).

The proposed approach is inspired by the popular early face recognition method *Fisherfaces* [1], where the inter-class scatter matrix is maximized, while the intra-class scatter matrix is minimized. Therefore, we call our method *deep Fisher faces*.

1.1 Related work

There has been a lot of activity regarding loss functions for face recognition and verification. Using an inter-class term in the loss has been proposed before, e.g. in the range loss [21] and in [3]. In the latter the authors explore a way to directly translate the Linear Discriminant Analysis (LDA) [4] into a training criterion for deep neural networks. However, in contrast to this we intend to directly extend the center loss as proposed [17], which means defining the inter-class term in dependence of the global centers defined by the center-loss and the input of the current batch.

Other loss functions include the contrastive loss [5] and the triplet loss [13], both aim to minimize distances of features of the same class while maximizing distances of features of different classes. This type of loss can be used either stand-alone[13], jointly with a softmax cross-entropy loss [14], or to finetune a baseline model [11] (e.g. trained with softmax cross-entropy).

Recently, also extensions of the softmax cross-entropy loss have been proposed by adding an $L2$ constraint [12, 16]. As the authors demonstrate, these losses can also be used jointly with other loss functions such as the the center loss or the loss used in this work.

2 Recap: The Center Loss

First we give a quick review of the center loss [17]. The aim of this loss is to minimize the distance of the face images to their class centers. This means the intra-class distance is minimized. Let m be the batch-size and t the current iteration. The authors of [17] define the center loss for the output of the underlying network x_i^t with labels y_i^t as (notation differs slightly from [17])

$$L_c = \frac{1}{2}\sum_{i=1}^{m} \|x_i^t - c_{y_i^t}^{t-1}\|_2^2 \tag{1}$$

where c_y^{t-1} is the center of class y after being updated using batch $t-1$. A model is trained using this loss with the standard backpropagation algorithm which uses the following gradient

$$\frac{\partial L_c}{\partial x_i^t} = x_i^t - c_{y_i^t}^{t-1} \tag{2}$$

The centers are updated with a specific learning rate α using the rule

$$c_y^t = c_y^{t-1} - \alpha \Delta c_y^{t-1} \tag{3}$$

with

$$\Delta c_y^{t-1} = \frac{\sum_{i=1}^{m} \delta(y_i^t = y)(c_{y_i^t}^{t-1} - x_i^t)}{1 + \sum_{i=1}^{m} \delta(y_i^t = y)} \tag{4}$$

The function $\delta(y = y\prime)$ returns 1 if y equals $y\prime$ and 0 otherwise and the learning rate α is set independently of the learning rate used for backpropagation. The updates of the centers are done after each iteration. Formula (3) can be seen as the formula to iteratively compute a mean, with the addition of the weight α. According to [17] the weight is necessary to prevent too much fluctuation in the updates due to the batch processing, since each batch only contains a small sample of the training set.

3 Deep Fisher faces

Inspired by the *Fisherfaces* [1] we want to augment the intra-class term of the center loss with an inter-class term that maximizes the distances between different classes. This can be achieved by including

$$\sum_{(y,y\prime)} \|c_y - c_{y\prime}\|_2^2 \tag{5}$$

where y and $y\prime$ are two different classes. However, for this term (Formula (5)) to be able to directly contribute to the training of the parameters of the underlying model using a gradient-based method such as backpropagation, the term needs to be somehow dependent on the model output x_i^t of the current iteration t. Such a dependency is not given in Formula (1), since the centers are updated only after the loss is evaluated. During the computation of the loss, the centers c_y^{t-1} are used, which have been computed in the previous iteration. For this reason we first modify the center loss by doing the updates of the centers first.

3.1 Modified center loss

Updating the centers first before the loss is evaluated simply means changing $c_{y_i^t}^{t-1}$ to $c_{y_i^t}^{t}$ in Formula (1) leading to

$$L_{mc} = \frac{1}{2} \sum_{i=1}^{m} \|x_i^t - c_{y_i^t}^t\|_2^2 \tag{6}$$

However, this change creates a functional dependence between the centers used in the loss at iteration t and x_i^t. This has to be considered during backpropagation. The new gradient is now given by

$$\frac{\partial L_{mc}}{\partial x_i^t} = (x_i^t - c_{y_i^t}^t)\left(1 - \frac{\partial c_{y_i^t}^t}{\partial x_i^t}\right) \tag{7}$$

$$= (x_i^t - c_{y_i^t}^t)\left(1 - \frac{\partial}{\partial x_i^t}\left(c_{y_i^t}^{t-1} - \alpha \Delta c_{y_i^t}^{t-1}\right)\right) \tag{8}$$

$$= (x_i^t - c_{y_i^t}^t)\left(1 - \frac{\alpha}{1 + \sum_{i\prime=1}^{m} \delta(y_{i\prime}^t = y_i^t)}\right) \tag{9}$$

This leads to an interesting result if we assume $0 < \alpha < 1$, which is typically the case (e.g. we set α to 0.5 in our experiments, see Section 4). Apart from the updated centers, the difference between Formula (2) and Formula (9) is a weighting factor that is defined by how often a class is seen in the batch. Classes that are seen more often get a higher weight compared to classes seen less often.

3.2 Adding the inter-class loss

While the minimization of the center loss in Formula (1) is bounded by zero, the maximization of Formula (5) is not bounded. This could lead to a learning behavior where the inter-class distances increase with little regard to the intra-class distances. Therefore, we bound the optimization of the inter-class distance with a margin turning the maximization into a minimization, as it is done in the range loss [21], contrastive loss [5] or triplet loss [13]. This leads to the following loss function

$$L_f = \underbrace{\frac{1}{2}\sum_{i=1}^{m} \|x_i^t - c_{y_i^t}^t\|_2^2}_{\text{Intra-class loss } L_{mc}} + \underbrace{\frac{1}{2}\sum_{(y,y\prime)\in Y^t} \max\left(m - \|c_y^t - c_{y\prime}^t\|_2^2, 0\right)}_{\text{Inter-class loss } L_b} \tag{10}$$

where the set Y^t contains all pairs of classes seen in batch t. Depending on the batch-size, the set Y^t can become relatively large. E.g., for a batch-size of 128 we have in the worst case 8128 pairs. To save computation time, a random subset can be used instead.

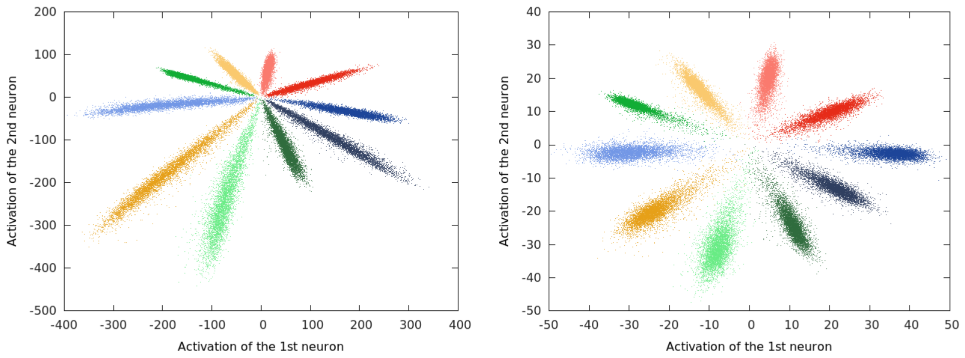
The gradient of the inter-class loss L_b is given by

$$\frac{\partial L_b}{\partial x_i^t} = \sum_{(y,y\prime)\in Y^t} \begin{cases} 0 & m \le \|c_y^t - c_{y\prime}^t\|_2^2 \\ -\frac{\partial}{\partial x_i^t}\left(\frac{1}{2}\|c_y^t - c_{y\prime}^t\|_2^2\right) & \text{otherwise} \end{cases} \tag{11}$$

where

$$\frac{\partial}{\partial x_i^t}\left(\frac{1}{2}\|c_y^t - c_{y\prime}^t\|_2^2\right) = \sum_{\substack{(y,y\prime)\in Y^t \\ y=y_i \vee y\prime=y_i}} \frac{\alpha(c_y^t - c_{y\prime}^t)}{1 + \sum_{i\prime=1}^{m} \delta(y_{i\prime}^t = y_i^t)} \begin{cases} 1 & y = y_i^t \\ -1 & y\prime = y_i^t \end{cases} \tag{12}$$

Using the center loss (c.f. Formula (1)) alone can lead to a state where the model maps all data-points to the same feature vector in the embedding space, as this would be an optimal solution (i.e. the center loss would be zero). For this reason the authors train the center loss



(a) Softmax CE (b) Softmax CE + center loss

Figure 1: Learned embedding space on MNIST.

jointly with a softmax cross-entropy loss. In the case of the deep Fisher faces the loss is high when the centers are very close, the gradients however are close to zero (see Formula (12)) making it difficult for the training algorithm to leave this state. Therefore, we also use the joint training with the softmax cross-entropy loss (L_{CE}) to achieve a stable learning procedure. As the center loss in [17] we weight the new loss with the hyper-parameter λ leading to the following over all loss function:

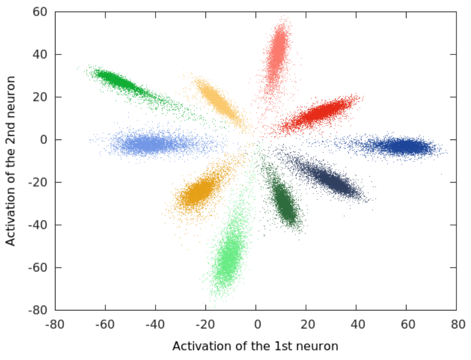
$$L_{mc} = L_{CE} + \lambda L_f \tag{13}$$

Similar to [21] it would be possible to introduce another hyper-parameter to weight the contribution of the inter-class part in L_f. However, in order to avoid the addition of another hyper-parameter we decided against this.

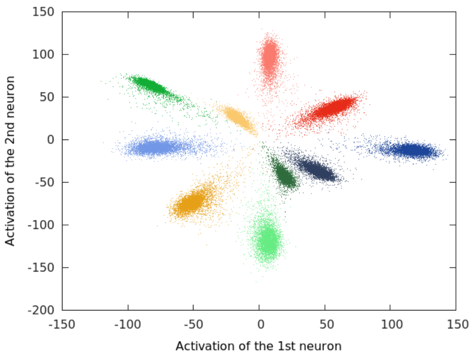
3.3 Example embeddings on MNIST

Before we evaluate the new loss on the face verification tasks, we illustrate the effect of the inter-class loss using the MNIST dataset [9]. Similar to [12, 17] we use a simple CNN with a bottleneck layer of size 2 as embedding layer to be able to visualize the embedding space without any additional methods. We train a baseline model with softmax cross-entropy and then finetune this model with the center loss and the deep Fisher faces with different margins. Apart from the margins, we use the same hyper-parameters for all models. Note that the plots are only meant to illustrate the effect of the different losses and we did not put an emphasis on recognition performance.

Figure 1 shows the embedding for the baseline model and the model finetuned by adding the center loss. By using the center loss the intra-class variance is reduced, but also the centers are closer in the embedding space (the range of the plot is reduced significantly). In contrast to this result the embeddings trained with deep Fisher faces (see Figure 2) show a relatively small intra-class variance, but with a higher inter-class variance. The higher margin also leads to a higher inter-class variance.



(a) Softmax CE + DFF with margin 1000



(b) Softmax CE + DFF with margin 5000

Figure 2: Learned embedding space on MNIST with deep Fisher faces.

4 Experimental evaluation

The evaluation on LFW [7] and YTF [18] is done with the same models that are not finetuned to the target database. We use the same model structure for all experiments. The model is a CNN based on the wide resnets introduced in [20]. The details are given in Table 1 using notation similar to [20] (e.g. conv1 $7 \times 7, 16$ describes a convolutional layer with kernel 7×7 and 16 feature maps). The blocks in group1 to group4 contain one dropout-layer in between. Overall, the model consists of roughly 17M learnable parameters. We use torch7 [2] to implement the deep Fisher faces and run the training and evaluation procedures described in the following sections.

4.1 Training

To train all our models we use the CASIA WebFace [19] database, which contains about 0.5 million images for a little over $10,000$ identities. We start by training a baseline model using only the softmax cross-entropy loss. With this baseline model as a constant starting point we then finetune the model with the center loss and the deep Fisher faces, respectively (in both cases trained jointly with softmax cross-entropy). The face images were center cropped to 224×224 RGB images. As in [17] we normalize the images by subtracting the value 127.5 from all pixels followed by a division by the value 128. To add more variation to the training data each image is flipped horizontally with a probability of 50%.

The training is done using the standard backpropagation with momentum set to 0.9 [8]. For regularization, the weight decay is set to 5×10^{-4} and the dropout ratio is set to 0.3. The weight initialization for the baseline model is done as in [6] and all biases are set to zero. We use a batch-size of 32 and an initial learning rate of 1×10^{-2}, which is gradually reduced.

For the two finetuned models we increase the batch-size to 128. We start with a learning rate of 1×10^{-3} and use constant values for λ and α throughout the experiments. The values are $\lambda = 0.003$ and $\alpha = 0.5$, the same as used in [17]. For the deep Fisher faces we sub-sample 128 pairs in each iteration and use a margin m is set to a value slightly above the average inter-class distance at the beginning of the finetuning.

The training runs on smaller GPUs with only 4GB memory.

Table 1: The wide resnet [20] model used in this paper. Each block consists of one or several convolutional layers in the resnet architecture. Each convolutional layer is followed by batch normalization and a ReLU activation function. The table shows the kernel-sizes and number of feature maps. For an input of 224 × 224 the model returns a 512-dimensional feature vector.

Name	Block type
conv1	$7 \times 7, 16$
max-pool	3×3
group1	$\begin{pmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{pmatrix} \times 2$
group2	$\begin{pmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{pmatrix} \times 2$
group3	$\begin{pmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{pmatrix} \times 2$
group4	$\begin{pmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{pmatrix} \times 2$
avg-pool	7×7

4.2 Labeled Faces in the Wild

The Labeled Faces in the Wild (LFW) [7] is a very popular benchmark for unconstrained face verification. It consists of 13233 images for 5749 identities in total. The images have been collected in unconstrained conditions as long as they were detectable by a face detector. This leads to a database with a wide range of variations, such as lighting, facial expression or pose. The evaluation is done using 10-fold cross validation with fixed splits provided by the database authors. Each split consists of 600 pairs, where half of them are positive pairs (same identity) and the other half are negative pairs (different identities). The database authors define several protocols that differ by what type of training data can be used. As most deep learning methods we use the unrestricted, labeled outside data protocol. We use the original center-cropped images as model input and receive a 512 dimensional feature representation as output. We do this for the original image and a horizontally flipped version and use the mean of the two resulting feature vectors as final representation. We do not perform any facial landmark detection or face alignment during the evaluation. The similarity scores are determined using the Euclidean distance.

The results are given in Table 2. We report the accuracy for our baseline model trained with softmax cross-entropy alone, the model finetuned by adding the center loss and the deep Fisher faces with the same fixed hyper-parameters. It is a bit surprising that the center loss achieves almost the same result as the baseline. We also tried other hyper-parameters for the center loss (different λ, α or longer training), but the results were always around 98%, with 98.1% as best. However, using the deep Fisher faces we gain a small improvement arriving at 98.4% accuracy. Note that this is achieved with a relatively small training set and very little preprocessing (e.g. no 2D alignment).

The comparison to the state-of-the-art methods shows that our results are competitive. We only include the results most relevant to our work, since there are too many to include them all. The gap to the best performing methods could be explained with our limited amount of data and lack of 2D alignment. Additionally, the deep Fisher faces could also be combined

Table 2: Results on LFW.

Method	Training images	2D alignment	Acc. [%]
VGG-Face [11]	2.6M	yes	99.0
FaceNet [13]	200M	no	98.9
FaceNet [13]	200M	yes	99.6
$L2$-constrained softmax [12]	3.7M	yes	99.8
Range loss [21]	1.5M	yes	99.5
Center loss [17]	0.7M	yes	99.3
Baseline	0.5M	no	97.9
Center loss	0.5M	no	98.0
Deep Fisher faces	0.5M	no	98.4

with the $L2$ softmax loss presented in [12], or an additional triplet embedding layer could be learned as it is done for VGG-Face [11] to boost performance further.

4.3 YouTube Faces

Similar to the LFW [7] database the YouTube Faces (YTF) [18] database is a popular benchmark for unconstrained face verification. However, instead of single still images, full videos are compared. In total there are 3425 videos for 1595 identities. On average, the videos consist of 181.3 frames. Again, the evaluation is done using a 10-fold cross validation where the splits are given with the database. Each split contains 250 pairs where the identity of both videos is the same and 250 pairs with different identities. We use the given bounding box data to crop the images, but expand the bounding box by a factor of 1.1. Again, we do not use facial landmarks or face alignment.

To compare two face videos it is a common approach to use the scores of a face or facial landmark detector to find the best frames in a video, which are then used for evaluation. However, since we do not have such scores available, we instead use the softmax operator as proposed in [10] in the context of template matching. Given two sets of images $P = x_1,...,x_p$ and $Q = x_1,...,x_q$, the similarity $s(P,Q)$ can be defined similar to [10]

$$s(P,Q) = \frac{1}{11} \sum_{\beta=-10}^{0} s_\beta(P,Q) \tag{14}$$

where

$$s_\beta(P,Q) = \frac{\sum\limits_{p\in P, q\in Q} s(x_p,x_q)e^{\beta s(x_p,x_q)}}{\sum\limits_{p\in P, q\in Q} e^{\beta s(x_p,x_q)}} \tag{15}$$

In [10] the two sets of images are given as templates (collection of still images and videos). Here we just have two videos to compare, so the sets are given by the frames. To save some computational complexity we random sample 128 frames per video. Another notable difference to [10] is that we use the Euclidean distance as similarity measure $s(x_p,x_q)$ between two images (consistent with the LFW evaluation) and therefore select β to be in the range of $[-10,...,0]$.

Table 3: Results on YTF.

Method	Training images	2D alignment	Acc. [%]
VGG-Face [11]	2.6M	yes	91.6
VGG-Face + Embedding learning [11]	2.6M	yes	97.3
FaceNet [13]	200M	yes	95.1
$L2$-constrained softmax [12]	3.7M	yes	96.1
Range loss [21]	1.5M	yes	93.7
Center loss [17]	0.7M	yes	94.9
Baseline	0.5M	no	90.4
Center loss	0.5M	no	90.9
Deep Fisher faces	0.5M	no	91.5

The results are given in Table 3. Again, with 91.5% the deep Fisher faces achieve a slightly better result than our center loss baseline.

5 Conclusion

In this paper we presented a novel loss function that extend the center loss [17] by an inter-class term. By this modification we obtain a loss that simultaneously maximizes inter-class distances while minimizing intra-class distances. The new loss is easy to implement and leads to an improved performance on both, LFW and YTF compared to our baseline center loss. It would be interesting to study the deep Fisher faces in combination with a normalization layer as it is used in the $L2$-constrained softmax loss [12]. Since the evaluation is done using normalized features it could be helpful to optimize the distances between the image features of a class to their centers as well as the distances between the centers directly in the normalized embedding space.

6 Acknowledgement

European Research Council
Established by the European Commission

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 694537). The work reflects only the authors' views and the European Research Council Executive Agency is not responsible for any use that may be made of the information it contains.

References

[1] Peter N. Belhumeur, João P Hespanha, and David J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on pattern analysis and machine intelligence*, 19(7):711–720, 1997.

[2] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, 2011.

[3] Matthias Dorfer, Rainer Kelz, and Gerhard Widmer. Deep linear discriminant analysis. *arXiv preprint arXiv:1511.04707*, 2015.

[4] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of human genetics*, 7(2):179–188, 1936.

[5] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, volume 2, pages 1735–1742. IEEE, 2006.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian 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.

[7] Gary B Huang, Manu Ramesh, Tamara Berg, and Erik 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.

[8] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[9] Yann LeCun, Corinna Cortes, and Christopher JC Burges. The mnist database of handwritten digits, 1998.

[10] Iacopo Masi, Anh Tuážěn Trážğn, Tal Hassner, Jatuporn Toy Leksut, and Gérard 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.

[11] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. In *BMVC*, volume 1, page 6, 2015.

[12] Rajeev Ranjan, Carlos D Castillo, and Rama Chellappa. L2-constrained softmax loss for discriminative face verification. *arXiv preprint arXiv:1703.09507*, 2017.

[13] Florian Schroff, Dmitry Kalenichenko, and James 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.

[14] Yi Sun, Ding Liang, Xiaogang Wang, and Xiaoou Tang. Deepid3: Face recognition with very deep neural networks. *arXiv preprint arXiv:1502.00873*, 2015.

[15] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior 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.

[16] Feng Wang, Xiang Xiang, Jian Cheng, and Alan L Yuille. Normface: l_2 hypersphere embedding for face verification. *arXiv preprint arXiv:1704.06369*, 2017.

[17] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In *European Conference on Computer Vision*, pages 499–515. Springer, 2016.

[18] Lior Wolf, Tal Hassner, and Itay 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.

[19] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z Li. Learning face representation from scratch. *arXiv preprint arXiv:1411.7923*, 2014.

[20] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[21] Xiao Zhang, Zhiyuan Fang, Yandong Wen, Zhifeng Li, and Yu Qiao. Range loss for deep face recognition with long-tail. *arXiv preprint arXiv:1611.08976*, 2016.