PRECISE ADJACENT MARGIN LOSS FOR DEEP FACE RECOGNITION

Xin Wei, Hui Wang, Bryan Scotney, and Huan Wan

School of Computing, Ulster University, BT370QB UK

ABSTRACT

Softmax loss is arguably one of the most widely used loss functions in CNNs. In recent years some Softmax variants have been proposed to enhance the discriminative ability of the learned features by adding additional margin constraints, which significantly improved the state-of-the-art performance of face recognition. However, the 'margin' referenced in these losses does not represent the real margin between the different classes in the training set. Furthermore, they impose a margin on all possible combinations of class pairs, which is unnecessary. In this paper we propose the Precise Adjacent Margin loss (PAM loss), which gives an accurate definition of 'margin' and has precise operations appropriate for different cases. PAM loss has better geometrical interpretation than the existing margin-based losses. Extensive experiments are conducted on LFW, YTF, MegaFace and FaceScrub datasets, and results show that the proposed method has state-of-the-art performance.

Index Terms— Margin, Loss, Deep learning, Convolutional neural networks, Face recognition.

1. INTRODUCTION

Over the past decade deep learning-based methods have made great strides in various areas of computer vision. Among these areas, the progress on face recognition is particularly remarkable because of the development of CNNs and associated loss functions.

In face recognition the loss functions with the best performance can be divided into two categories: loss functions based on Euclidean distance and loss functions based on cosine similarity. Most of the time, cosine similarity-based losses show better performance than Euclidean distance-based losses. Cosine similarity-based losses include L-Softmax loss [1], A-Softmax loss [2], AM-Softmax loss [3] and ArcFace loss [4]. These losses are all derived from the commonly used Softmax loss by adding margin constraints and applying feature or weight normalisation, and they have achieved state-of-the-art performance in deep face recognition. However, these loss functions have two common defects. Firstly, the 'margin' referenced in the above losses is the margin between the decision boundaries of Softmax, which do not represent the real margin between the different classes in the training set. Secondly, the above losses impose a margin on all possible combinations of the class pairs, which is unnecessary.

In this paper we propose the Precise Adjacent Margin loss (PAM loss), which gives 'margin' a meaning that represents the real margin between the different classes in the training set. Differently from the above losses, PAM loss optimises only the margin for a limited number of class pairs. Our main contributions are as follows: (a) We propose PAM loss to improve the discriminative ability of the deep features. To the best of our knowledge, PAM loss is the first loss that uses the real margin between the different classes in the training set. (b) To implement PAM loss, we propose a learning algorithm to obtain the range of each class (namely, the cosine similarity between the class centre and the class edge). (c) Extensive experiments are conducted on public datasets including LFW [5], YTF [6], MegaFace [7] and FaceScrub [8] datasets, and the results obtained verify the state-of-the-art performance of PAM loss.

2. RELATED WORK

Currently Softmax loss is the most widely used loss function in CNNs and is formulated as follows:

$$\mathcal{L}_S = -\frac{1}{N}\sum_{i=1}^{N} log\frac{e^{W_{y_i}^T f_i + b_{y_i}}}{\sum_{j=1}^{P} e^{W_j^T f_i + b_j}} \quad (1)$$

where N is the batch size, P is the number of classes in the entire training set, $f_i \in R^d$ is the feature vector of the ith sample in the mini-batch, y_i is the class label of the ith sample, $W_j \in R^d$ is the jth column of the weight matrix W in the final fully connected layer and b_j represents the bias term of the jth class[1]. Considering the definition of cross entropy, we can find from Eq(1) that Softmax loss is actually the cross entropy between the predicted label and the true label, which means that Softmax loss focuses only on the correctness of classification. Hence, the purpose of Softmax loss is to separate samples of different classes, rather than learning discriminative features.

To enhance the discriminative ability of the features, L-Softmax loss, A-Softmax loss, AM-Softmax loss and ArcFace loss have been proposed successively over the past two

[1]To save space, we just define the mathematical symbols once and then use them following the initial definition without additional explanation.

years. Eq.(2) and Eq.(3) show the formulation of L-Softmax loss and A-Softmax loss, respectively:

$$\mathcal{L}_L = -\frac{1}{N}\sum_{i=1}^{N} log\frac{e^{\|W_{y_i}\|\|f_i\|\psi(\theta_{y_i})}}{e^{\|W_{y_i}\|\|f_i\|\psi(\theta_{y_i})} + \sum_{j=1,j\neq y_i}^{P} e^{\|W_j\|\|f_i\|\psi(\theta_j)}} \tag{2}$$

$$\mathcal{L}_A = -\frac{1}{N}\sum_{i=1}^{N} log\frac{e^{\|f_i\|\psi(\theta_{y_i})}}{e^{\|f_i\|\psi(\theta_{y_i})} + \sum_{j=1,j\neq y_i}^{P} e^{\|f_i\|\psi(\theta_j)}} \tag{3}$$

where $\psi(\theta_{y_i})$ equals $(-1)^k cos(m\theta_{y_i}) - 2k, \theta_{y_i} \in (\frac{k\pi}{m}, \frac{(k+1)\pi}{m})$, $k \in (0, m-1)$, and m is the size of margin. Here $m \geq 1$, which is used to adjust the target angular margin. Based on the original Softmax loss in Eq.(1), L-Softmax loss and A-Softmax loss modify the FC layer formulation from $W_{y_i}^T f_i + b_{y_i}$ to $\|W_{y_i}\|\|f_i\|cos\theta_{y_i}$ by setting the bias b_{y_i} to 0. As a result, the distance measurement is transferred from the Euclidean distance to the cosine similarity. Differently from L-Softmax loss, L2 weight normalisation is applied in A-Softmax loss by setting $\|W_{y_i}\| = 1$. Since m is introduced as a multiplier on θ_{y_i}, the corresponding margin is called multiplicative angular margin.

On the basis of A-Softmax, AM-Softmax adopts L2 feature normalisation and replaces $\psi(\theta_{y_i})$ with $cos(\theta_{y_i}) - m$, where m is called additive cosine margin:

$$\mathcal{L}_{AM} = -\frac{1}{N}\sum_{i=1}^{N} log\frac{e^{s(cos(\theta_{y_i})-m)}}{e^{s(cos(\theta_{y_i})-m)} + \sum_{j=1,j\neq y_i}^{P} e^{scos(\theta_j)}}. \tag{4}$$

$\|f_i\|$ is set by L2 normalisation and is re-scaled to s. After AM-Softmax, Deng et al. [4] proposed ArcFace loss, which further updates $cos(\theta_{y_i}) - m$ by $cos(\theta_{y_i} + m)$. Even though ArcFace still uses the additive margin, it has better geometric meaning as m corresponds directly to the angle of the margin.

3. THE PROPOSED PAM LOSS

Fig. 1 provides a 2D visualisation of the aforementioned loss functions in the case of two classes. The blue and green areas represent the target areas of the two classes, respectively. From the formulation of Softmax loss and the aforementioned variants (including L-Softmax, A-Softmax, AM-Softmax and ArcFace), it can be inferred that the target area is determined by W_j in Softmax loss and by W_j and m in the variants. The geometrical interpretation of Softmax loss is shown in Fig. 1(a), where $W_1^T d_0 = W_2^T d_0$. Therefore, class 1 and class 2 share the same decision boundary d_0 and there is no margin between the target areas of class 1 and class 2. Fig. 1(b) illustrates the case of the variants based on Softmax and margin constraint. d_1 and d_2 are the corresponding decision boundaries of class 1 and class 2 and there is a margin between the target area of the two classes. The size of the margin is determined by m. For example, if the variant is AM-Softmax, $W_1^T d_1 - m = W_2^T d_1$ and $W_2^T d_2 - m = W_1^T d_2$. If the variant is ArcFace loss, $\|W_1\|\|d_1\|cos(\theta_1 + m) = \|W_2\|\|d_1\|cos\theta_2$. Since $\|W_1\| = \|d_1\| = 1$, $cos(\theta_1 + m) = cos\theta_2$ and $m = \theta_2 - \theta_1$, which is the angle of the margin.

However, the target area of a class is not its real area. Its real area of a class is determined by the samples of the class in the training set. Ideally, the real area is expected to converge to the target area. However, during training, the real area could be smaller or larger than the target area or even has no overlap with the target area; the latter is common at the early stage of training. A simple example is shown in Fig. 1(c). In the training process, the parameters in CNNs are learned by the class distribution information in the training set. In other words, the feedback from the training set guides the parameters in CNNs to update. Therefore, our approach is motivated by the view that the feedback information should be as precise as possible and so we try to introduce the real margin into the loss function.

The aforementioned variants impose a margin on all possible combinations of the class pairs. This approach is simple and convenient but unnecessary. The original purpose of training is to separate the overlapping classes. The second purpose is to enlarge the margin between those classes which are close to each other. For those classes which are already far from each other, there is no need to optimise the parameters to separate them even farther (as they have satisfied the requirement of classification). As the expressive power of a neural network is not unlimited, optimising the parameters to satisfy the requirements for some of the classes will inevitably have an influence on the distribution of other classes.

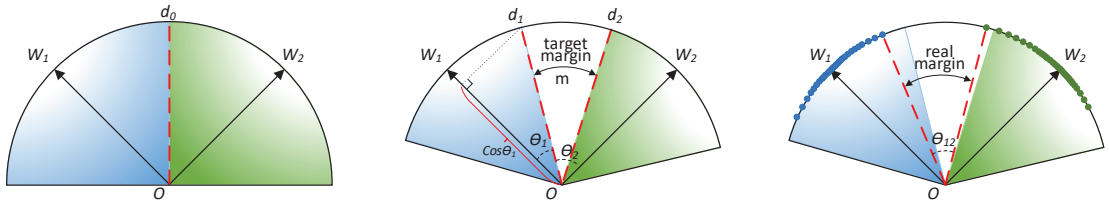
Motivated by the considerations above, we propose the Precise Adjacent Margin loss (PAM loss). PAM loss is used along with the AM-Softmax loss (see Eq. (5)) and has two versions, whose formulations are shown in Eq. (6) and Eq. (7), respectively.

$$\mathcal{L} = \mathcal{L}_{AM} + \lambda\mathcal{L}_P \tag{5}$$

$$\begin{aligned} &\mathcal{L}_{P\_v1} = \frac{\sum_{Top}(S,P)}{P} \\ &S = \{\varphi(cos(\theta_{ij})) : i,j = 1,2,3,...,P; i > j\} \end{aligned} \tag{6}$$

$$\begin{aligned} &\mathcal{L}_{P\_v2} = \frac{\sum_{i=1}^{P}\sum_{Top}(S_i,2)}{2P} \\ &S_i = \{\varphi(cos(\theta_{ij})) : j = 1,2,3,...,P\} \end{aligned} \tag{7}$$

where λ is the hyper-parameter for adjusting the impact of introducing the PAM loss, θ_{ij} is the real margin angle between class i and class j as illustrated in Fig. 1(c). $\varphi(cos(\theta_{ij})) = cos(\theta_{ij})$ if $\theta_{ij} > 0$. When $\theta_{ij} \leq 0$, $\varphi(cos(\theta_{ij})) = -cos(\theta_{ij}) + 2$, which adds more penalty to the overlapping classes and ensures the continuity of $\varphi(cos(\theta_{ij}))$. S and S_i are sets of values of $\varphi(cos(\theta_{ij}))$. $\sum_{Top}(S,P)$ denotes the sum of the P largest elements in set S. Both versions of PAM loss aim to optimise the margins between different classes. The ideal approach would optimise the margins of all the adjacent classes. However, it is extremely time-consuming to select out all these adjacent classes in the hypersphere. In PAM loss v1, a conservative strategy is adopted, which penalises P pairs of classes that have the largest $\varphi(cos(\theta_{ij}))$ values. This is because the minimum number of pairs of the adjacent classes is P, which



(a) Softmax with Weight and Feature Normalisation. (b) Variants based on Softmax and Margin Constraint. (c) Interpretation of the Real Margin.

Fig. 1. Geometrical interpretation of different losses. The three sub-figures show the case of 2D feature space. The blue and green areas represent the target areas of two classes, respectively. W_1 and W_2 are the corresponding weights of class 1 and class 2 in the FC layer. As described in Fig. 6 of [9], W_1 and W_2 will converge to the centres of their corresponding classes, and therefore W_1 and W_2 can be regarded as the approximate centres of class 1 and class 2, respectively. In (a), d_0 is the decision boundary between class 1 and class 2. In (b), d_1 and d_2 are the corresponding decision boundaries of class 1 and class 2.

occurs when all the classes line up in a circle on the surface of the hypersphere. In PAM loss v2, we pay attention to every class. PAM loss v2 finds the nearest neighbour class of each class and penalises the margin between them.

Calculating $cos(\theta_{ij})$ is the key aspect of implementing the PAM loss. To calculate $cos(\theta_{ij})$, two parts are needed: the class centre and the cosine range of the class, where the cosine range of the class means the cosine similarity between the class centre and the farthest sample of the class. As training goes on, W_j gradually converges to the centre of class j ($j = 1, 2, ..., P$) [9]. W_j is easily obtained from the FC layer, so we use W_j as the approximation of the centre of the jth class. For the cosine range of class j we propose the following learning algorithm to update it recursively. $R(j)$ is initialised to 1 and is then updated using the following iterations:

$$R(j)^{t+1} = R(j)^t + \sum_{i=1}^{N} \phi(y_i, j) \cdot \Delta R_i, j = 1, 2, ..., P. \quad (8)$$

$$\Delta R_i = \begin{cases} cos(W_{y_i}, f_i) - R(y_i)^t, & R(y_i)^t > cos(W_{y_i}, f_i) \\ \beta \cdot (cos(W_{y_i}, f_i) - R(y_i)^t), & R(y_i)^t \leq cos(W_{y_i}, f_i) \end{cases} \quad (9)$$

where $\phi(y_i, j) = 1$ if $y_i = j$, $\phi(y_i, j) = 0$ if $y_i \neq j$, β is named shrink rate which is used to adjust the shrink speed of the class range. Eq. (9) contains two cases: (a) if the cosine similarity between the input sample and the corresponding class centre is less than the current recorded class range, the class range is replaced directly by their cosine similarity; (b) otherwise, the class range shrinks by the product of β and their cosine similarity. Case (a) keeps the range of the class up to date, but as training goes on, the real class range tends to become smaller and smaller. Therefore, case (b) is designed to help the learned class range shrink to the real value.

With the class centre and the cosine range of the class, $cos(\theta_{ij})$ can be calculated. Let $R(i) = cos(\theta_i)$, $R(j) = cos(\theta_j)$, and $W_i \cdot W_j = cos(\theta)$, then $cos(\theta_{ij}) = cos(\theta - \theta_i - \theta_j)$. By solving this equation, we get:

$$\begin{aligned} cos(\theta_{ij}) =& W_i W_j R(i) R(j) - W_i W_j (\sqrt{(1 - R(i)^2)(1 - R(j)^2)} \\ &+ \sqrt{(1 - (W_i W_j)^2)(1 - R(i)^2)} R(j) \\ &+ \sqrt{1 - (W_i W_j)^2} R(i) R(j). \end{aligned} \quad (10)$$

In our implementation, we do a more efficient one-time calculation to obtain all $cos(\theta_{ij})$ values ($i, j = 1, 2, 3, ..., P; i > j$) by matrix manipulation between W and $[R(1), ..., R(P)]$, where W is the weight matrix in the final FC layer.

Table 1. Parameter settings for the training and the testing.

Parameter	Value	Parameter	Value
batch size	120	moving average decay	0.9999
image size	160*160	AM-Softmax scalar	40.0
epoch size:	1000	AM-Softmax margin	0.3
embedding size	512	PAM loss shrink rate	0.01
random flip	True	LR of epoch 0∼99	0.05
keep probability	0.4	LR of epoch 100∼199	0.005
optimizer	ADAM	LR of epoch 200∼360	0.0005
weight decay	0.0005	Note: LR denotes learning rate.	

4. EXPERIMENTS

4.1. Implementation Details

We implement four schemes with Tensorflow[2] by combining Inception-ResNet-v1 [10] with different loss functions: ResNet + Softmax, ResNet + AM-Softmax, ResNet + AM-Softmax + PAM loss v1, and ResNet + AM-Softmax + PAM loss v2. For convenience, we use "Softmax", "AM-Softmax", "PAM loss v1" and "PAM loss v2" to represent these four schemes, respectively, in the experimental results.

VGGFace2 [11] is used as the training set in all experiments. We removed the face images in VGGFace2 that might overlap with the benchmark testing sets to ensure the reliability of the experimental results. The resulting training set consists of 3.05 million facial images from more than 8,000 identities. For all face datasets used in our experiments, we apply MTCNN [12] for face detection. If MTCNN detection fails on a training image, we just remove the image from the training set. If MTCNN fails on a testing image, we use the landmarks or the bounding boxes provided by the authorities. As the learned class range is not steady in the early stage, the hyper parameter λ is set to 0 in the first 275 epochs. After that, we manually optimise λ. Since it is not very sensitive to the performance, we just try multiple different values on each testing set and choose the value that leads to the best result.

[2]https://www.tensorflow.org/

Table 2. Verification accuracy of state-of-the-art methods on LFW and YTF datasets.

Methods	Images	LFW	YTF
ICCV17' Range Loss [13]	1.5M	99.52	93.7
CVPR17' Marginal Loss [14]	4M	99.48	96.0
CVPR15' DeepID2+ [15]		99.47	93.2
CVPR14' Deep Face [16]	4M	97.35	91.4
CVPR15' Fusion [17]	500M	98.37	
ICCV15' FaceNet [18]	200M	99.63	95.1
ECCV16' Centre Loss [19]	0.7M	99.28	94.9
NIPS16' Multibatch [20]	2.6M	98.20	
ECCV16' Aug [21]	0.5M	98.06	
ICML16' L-Softmax [1]	0.5M	98.71	
CVPR17' A-Softmax [2]	0.5M	99.42	95.0
Softmax	3.05M	99.50	95.22
AM-Softmax	3.05M	99.57	95.62
PAM loss v1	3.05M	99.63	96.14
PAM loss v2	3.05M	99.62	96.00

The detailed parameter settings for the training and the testing are shown in Table 1.

4.2. Results on LFW and YTF

We have compared the proposed PAM loss with the state-of-the-art methods[3] on two benchmark datasets – LFW [5] and YTF [6]. In our experiments we follow the standard experimental protocol of "unrestricted with labelled outside data" [22]. Table 2 shows the results of the proposed methods and the state-of-the-art methods on LFW and YTF datasets, from which we can observe the following. On LFW, both versions of PAM loss outperform the related methods: Softmax, L-Softmax, A-Softmax and AM-Softmax. FaceNet has the same accuracy as the proposed PAM loss v1. However, FaceNet [18] uses 200 million images for training whilst the PAM loss only uses 3.05 million images for training. Compared with the other state-of-the-art methods, PAM loss has the highest verification accuracy. On YTF dataset, PAM loss v1 has an accuracy of 96.14%, which is higher than all the other methods. PAM loss v2 is similar on accuracy to Marginal loss, but Marginal loss uses a larger training set and has poorer performance on LFW. The results on LFW and YTF datasets demonstrate the effectiveness and the state-of-the-art performance of the proposed methods.

4.3. MegaFace Challenge 1 on FaceScrub

In this section, experiments are conducted on the MegaFace dataset [7] and the FaceScrub dataset [8]. We follow the experimental protocol of MegaFace Challenge 1, where MegaFace is set as the distractor set and FaceScrub is set as the testing set. The evaluation code [7] is provided by the MegaFace team. More details about the experimental protocol can be found in [7]. Fig. 2 and Fig. 3 show the CMC curves and the ROC curves with 1 million distractors on MegaFace Set 1, respectively. The results of the benchmark methods (including Barebones FR, SIAT MMLAB, Vocord and Faceall) are generated from the features provided by the MegaFace team[4]. It can be seen from Fig. 2 and Fig. 3 that PAM loss v1 and PAM loss v2 have better identification and verification performance than Softmax, AM-Softmax, and the other benchmark methods. PAM loss v2 outperforms PAM loss v1 in both figures, which indicates that PAM loss v2 has stronger ability in the case of 1 million distractors. The results on the MegaFace and the FaceScrub datasets confirm the effectiveness of the proposed methods.

[3] ArcFace [4] is not included in Table 2 as it hasn't been formally published, which means it hasn't passed the peer review. Additionally, after reviewing the comments on the public codes of Arcface, we found that its experimental methods and the results were still controversial.

Fig. 2. The CMC curves of different methods with 1 million distractors on MegaFace Set 1.

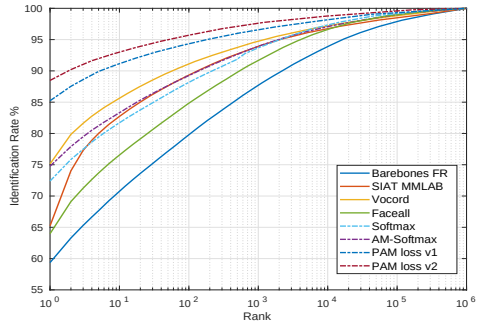
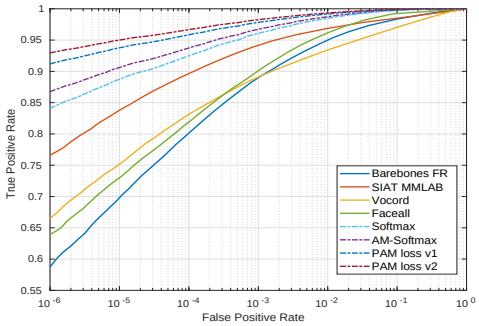


Fig. 3. The ROC curves of different methods with 1 million distractors on MegaFace Set 1.



5. CONCLUSION

In this paper, we have proposed the PAM loss, which gives 'margin' a meaning that represents the real margin between the different classes in the training set. To implement PAM loss, we also propose a learning algorithm to obtain the range of each class. Extensive experiments are conducted on LFW [5], YTF [6], MegaFace [7] and FaceScrub [8] datasets. Results demonstrate the effectiveness of the proposed methods and confirm the state-of-the-art performance of PAM loss.

[4] The download link of features provided by MegaFace team: http://megaface.cs.washington.edu/participate/challenge.html

6. REFERENCES

[1] Weiyang Liu, Yandong Wen, Zhiding Yu, and Meng Yang, "Large-Margin Softmax Loss for Convolutional Neural Networks," in *ICML*, 2016, pp. 507–516.

[2] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song, "SphereFace: Deep Hypersphere Embedding for Face Recognition," in *CVPR*, Honolulu, HI, 2017, pp. 6738–6746, IEEE.

[3] F. Wang, J. Cheng, W. Liu, and H. Liu, "Additive Margin Softmax for Face Verification," *IEEE Signal Processing Letters*, vol. 25, no. 7, pp. 926–930, 2018.

[4] Jiankang Deng, Jia Guo, and Stefanos Zafeiriou, "ArcFace: Additive Angular Margin Loss for Deep Face Recognition," *arXiv:1801.07698 [cs]*, 2018, arXiv: 1801.07698.

[5] 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," Tech. Rep. 07-49, University of Massachusetts, Amherst, 2007.

[6] Lior Wolf, Tal Hassner, and Itay Maoz, "Face recognition in unconstrained videos with matched background similarity," in *CVPR*, 2011, pp. 529–534.

[7] Ira Kemelmacher-Shlizerman, Steven M Seitz, Daniel Miller, and Evan Brossard, "The megaface benchmark: 1 million faces for recognition at scale," in *CVPR*, 2016, pp. 4873–4882.

[8] Hong-Wei Ng and Stefan Winkler, "A data-driven approach to cleaning large face datasets," in *ICIP*, 2014, pp. 343–347.

[9] Feng Wang, Xiang Xiang, Jian Cheng, and Alan Loddon Yuille, "NormFace: L $_2$ Hypersphere Embedding for Face Verification," in *Proceedings of the 2017 ACM on Multimedia Conference - MM '17*, Mountain View, California, USA, 2017, pp. 1041–1049, ACM Press.

[10] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke, "Inception-v4, inception-resnet and the impact of residual connections on learning," *CoRR*, vol. abs/1602.07261, 2016.

[11] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman, "Vggface2: A dataset for recognising faces across pose and age," *arXiv preprint arXiv:1710.08092*, 2017.

[12] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499–1503, 2016.

[13] X. Zhang, Z. Fang, Y. Wen, Z. Li, and Y. Qiao, "Range Loss for Deep Face Recognition with Long-Tailed Training Data," in *ICCV*, 2017, pp. 5419–5428.

[14] Jiankang Deng, Yuxiang Zhou, and Stefanos Zafeiriou, "Marginal loss for deep face recognition," in *CVPR, Faces "in-the-wild" Workshop/Challenge*, 2017, vol. 4.

[15] Y. Sun, X. Wang, and X. Tang, "Deeply learned face representations are sparse, selective, and robust," in *CVPR*, 2015, pp. 2892–2900.

[16] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf, "DeepFace: Closing the Gap to Human-Level Performance in Face Verification," in *CVPR*, 2014, pp. 1701–1708.

[17] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Web-scale training for face identification," in *CVPR*, 2015, pp. 2746–2754.

[18] Florian Schroff, Dmitry Kalenichenko, and James Philbin, "Facenet: A unified embedding for face recognition and clustering," in *CVPR*, 2015, pp. 815–823.

[19] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao, "A Discriminative Feature Learning Approach for Deep Face Recognition," in *ECCV*, 2016, pp. 499–515.

[20] Oren Tadmor, Tal Rosenwein, Shai Shalev-Shwartz, Yonatan Wexler, and Amnon Shashua, "Learning a Metric Embedding for Face Recognition Using the Multibatch Method," in *NIPS*, 2016, pp. 1396–1397.

[21] Iacopo Masi, Anh Tuan Tran, Tal Hassner, Jatuporn Toy Leksut, and Gérard Medioni, "Do We Really Need to Collect Millions of Faces for Effective Face Recognition?," in *ECCV*, 2016, pp. 579–596.

[22] Gary B Huang and Erik Learned-Miller, "Labeled faces in the wild: Updates and new reporting procedures," *Dept. Comput. Sci., Univ. Massachusetts Amherst, Amherst, MA, USA, Tech. Rep*, pp. 14–003, 2014.