# Additive Margin Softmax for Face Verification

Feng Wang, Jian Cheng, Weiyang Liu, and Haijun Liu

***Abstract*—In this letter, we propose a conceptually simple and intuitive learning objective function, i.e., additive margin softmax, for face verification. In general, face verification tasks can be viewed as metric learning problems, even though lots of face verification models are trained in classification schemes. It is possible when a large-margin strategy is introduced into the classification model to encourage intraclass variance minimization. As one alternative, angular softmax has been proposed to incorporate the margin. In this letter, we introduce another kind of margin to the softmax loss function, which is more intuitive and interpretable. Experiments on LFW and MegaFace show that our algorithm performs better when the evaluation criteria are designed for very low false alarm rate.**

***Index Terms*—Deep learning, face verification, metric learning.**

## I. Introduction

FACE verification is widely used for identity authentication in numerous areas, such as finance, military, public security, etc. Nowadays, most face verification models are built upon deep convolutional neural networks and supervised by classification loss functions [11], [20], [21], [23] or metric learning loss functions [15], [18], [19]. Metric learning loss functions usually require sample mining strategies and the final performance largely depends on these strategies, so increasingly more researchers focus their attentions on improving classification loss functions [11], [21], [23].

Current prevailing classification loss functions are mostly based on the widely used softmax loss. The softmax loss is good at optimizing the interclass variance, but is unable to reduce the intraclass variation. To address this, many new loss functions are proposed to minimize the intraclass variation. In [23], Wen *et al.* proposed to add a regularization term to penalize the feature-to-center distances. In [14], [17], [21], researchers proposed to use a scale parameter to control the "temperature" [3] of the softmax loss, producing higher gradients to the well-separated samples to further shrink the intraclass variance. In [11] and [12], Liu *et al.* introduced an angular margin to push the classification boundary closer to the weight vector of each class. Liu *et al.* [11] also provided a theoretical guidance of training a metric learning model using classification loss functions.

To better understand our algorithm, we will first give a brief review of the original softmax and the angular softmax (A-Softmax) loss [11]. The formulation of the original softmax loss is given by

$$\begin{aligned}\mathcal{L}_S &= -\frac{1}{n}\sum_{i=1}^{n}\log\frac{e^{W_{y_i}^T\mathbf{f}_i}}{\sum_{j=1}^{c}e^{W_j^T\mathbf{f}_i}}\\ &= -\frac{1}{n}\sum_{i=1}^{n}\log\frac{e^{\|W_{y_i}\|\|\mathbf{f}_i\|\cos(\theta_{y_i})}}{\sum_{j=1}^{c}e^{\|W_j\|\|\mathbf{f}_i\|\cos(\theta_j)}}\end{aligned} \tag{1}$$

where $f$ is the input of the last fully connected layer, $W_j$ is the $j$th column of the last fully connected layer. $W_{y_i}^T\mathbf{f}_i$ is also called as the target logit [16] of the $i$th sample.

In the A-Softmax loss, the authors proposed to normalize the weight vectors (making $\|W_i\|$ to be 1) and generalize the target logit from $\|\mathbf{f}_i\|cos(\theta_{y_i})$ to $\|\mathbf{f}_i\|\psi(\theta_{y_i})$

$$\mathcal{L}_{\text{AS}} = -\frac{1}{n}\sum_{i=1}^{n}\log\frac{e^{\|\mathbf{f}_i\|\psi(\theta_{y_i})}}{e^{\|\mathbf{f}_i\|\psi(\theta_{y_i})}+\sum_{j=1,j\neq y_i}^{c}e^{\|\mathbf{f}_i\|\cos(\theta_j)}} \tag{2}$$

where $\psi(\theta)$ is usually a piecewise function defined as follows:

$$\begin{aligned}\psi(\theta) &= \frac{(-1)^k\cos(m\theta)-2k+\lambda\cos(\theta)}{1+\lambda}\\ \theta &\in \left[\frac{k\pi}{m},\frac{(k+1)\pi}{m}\right]\end{aligned} \tag{3}$$

where $m$ is an integer larger than 1 and $\lambda$ is a hyperparameter to control how hard the classification boundary should be pushed. During training, $\lambda$ is annealing from 1000 to a small value to make the angular space of each class more and more compact. In their experiments, they set the minimum value of $\lambda$ to be 5 and $m = 4$, which is approximately $m = 1.5$ (see Fig. 1).

In this letter, we propose a novel and more interpretable way to import the angular margin into the softmax loss. We formulate an additive margin via $\cos\theta - m$, which is simpler than [11] and yields better performance. From (3), we can see that $m$ is multiplied to the target angle $\theta_{y_i}$, so this type of margin is incorporated in a multiplicative manner. Since our margin is a scalar subtracted from the cos$\theta$, we call our loss function additive margin softmax (AM-Softmax). Another recent additive margin form is described in [8], which is based on the original softmax, whereas ours is based on NormFace [21]. Recently

Manuscript received January 9, 2018; revised March 19, 2018; accepted March 23, 2018. Date of publication April 4, 2018; date of current version May 21, 2018. This work was supported in part by the National Natural Science Foundation of China under Grant 61671125, Grant 61201271, and Grant 61301269; and in part by the State Key Laboratory of Synthetical Automation for Process Industries (PAL-N201401). The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Phillip Ainsleigh. *(Corresponding author: Jian Cheng.)*

F. Wang, J. Cheng, and H. Liu are with the Department of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: feng.wff@gmail.com; chengjian@uestc.edu.cn; haijun_liu@126.com).

W. Liu is with the College of Computing, Georgia Institute of Technology, Atlanta, GA 30332, USA (e-mail: wyliu@gatech.edu).

Color versions of one or more of the figures in this letter are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/LSP.2018.2822810

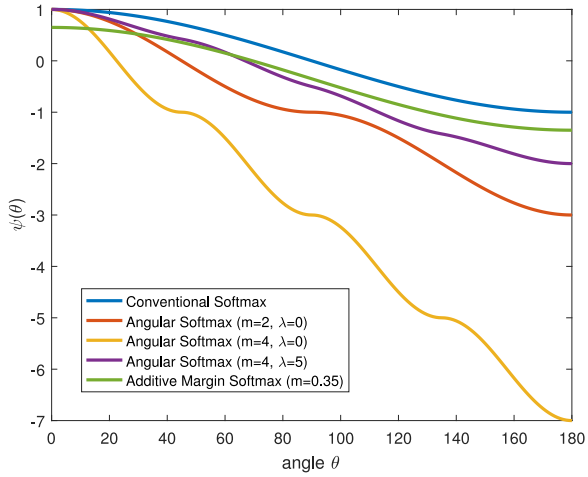


Fig. 1. $\psi(\theta)$ for conventional softmax, A-Softmax [11] and our proposed hard margin softmax. For A-Softmax, we plot the logit curve for three parameter sets. From the curves we can infer that $m = 4, \lambda = 5$ lies between conventional softmax and A-Softmax with $m = 2, \lambda = 0$, which means it is approximately $m = 1.5$. Our proposed AM-Softmax with optimized parameter $m = 0.35$ is also plotted and we can observe that it is similar with A-Softmax with $m = 4, \lambda = 5$ in the range $[0°, 90°]$, in which most of the real-world $\theta$s lie.

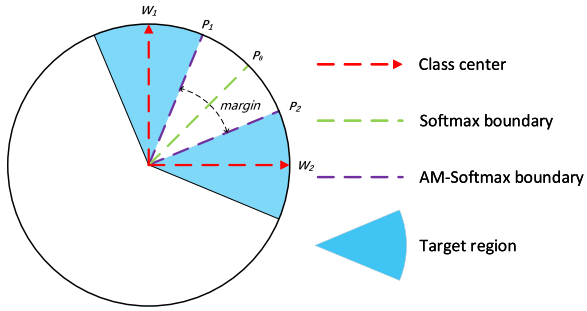


Fig. 2. Conventional softmax's decision boundary and AM-Softmax's decision boundary. For conventional softmax, the decision boundary is at $P_0$, where $W_1^T P_0 = W_2^T P_0$. For AM-Softmax, the decision boundary for class 1 is at $P_1$, where $W_1^T P_1 - m = W_2^T P_1 = W_1^T P_2$. Note that the distance marked on this figure does not represent the real distances. The real distance is a function of the cosine of the angle, whereas in this figure, we use the angle as the distance for better visualization effect. Here we use the word "center" to represent the weight vector of the corresponding class in the last inner-product layer, even though they may not be exactly the mean vector of the features in the class. The relationship between the weight vector "agent" and the features' mean vector "center" is described in [21, Fig. 6].

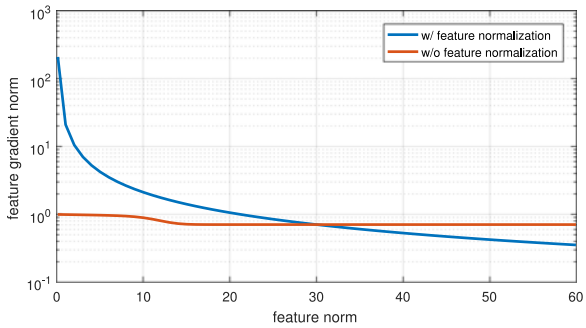


Fig. 3. Feature gradient norm w.r.t. the feature norm for softmax loss with and without feature normalization. The gradients are calculated using the weights from a converged network. The feature direction is selected as the mean vector of one selected target center and one nearest class center. Note that the $y$-axis is in logarithmic scale for better visualization. For softmax loss with feature normalization, we set $s = 30$. That is why the intersection of these two curves is at 30.

two concurrent works [1], [22] also shows the effectiveness of additive margin based on softmax loss function.

Experiments on LFW BLUFR protocol [9] and MegaFace [6] show that our loss function achieves better results than the current state-of-the-art approaches on the evaluation criteria with very low false alarm rate (FAR), e.g., 1e−4 on BLUFR or 1e−6 on MegaFace.

## II. Additive Margin Softmax

In this section, we will first describe the definition of the proposed loss function. Then, we will discuss about the intuition and interpretation of the loss function.

### A. Definition

Compared with $\psi(\theta)$ defined in A-Softmax [11] (3), our definition is more simple and intuitive, given by

$$\psi(\theta) = \cos\theta - m. \tag{4}$$

During implementation, the input is actually $x = \cos\theta_{y_i} = \frac{W_{y_i}^T f_i}{\|W_{y_i}\| \|f_i\|}$, so the forward propagation is

$$\Psi(x) = x - m. \tag{5}$$

Since we use cosine as the similarity to compare two face features, we follow [14] and [21] to apply both feature normalization and weight normalization to the inner product layer in order to build a cosine layer. Then, we scale the cosine values using a hyperparameter $s$ as suggested in [14] and [21]. Finally, the loss function is

$$\begin{aligned}\mathcal{L}_{\text{AMS}} &= -\frac{1}{n}\sum_{i=1}^{n}\log\frac{e^{s\cdot(\cos\theta_{y_i}-m)}}{e^{s\cdot(\cos\theta_{y_i}-m)}+\sum_{j=1,j\neq y_i}^{c} e^{s\cdot\cos\theta_j}} \\ &= -\frac{1}{n}\sum_{i=1}^{n}\log\frac{e^{s\cdot(W_{y_i}^T \mathbf{f}_i-m)}}{e^{s\cdot(W_{y_i}^T \mathbf{f}_i-m)}+\sum_{j=1,j\neq y_i}^{c} e^{sW_j^T \mathbf{f}_i}}.\end{aligned} \tag{6}$$

In this letter, we assume that the norm of both $W_i$ and $\mathbf{f}$ are normalized to 1 if not specified. In [21], Wang *et al.* propose to let $s$ to be learned through backpropagation. However, after the margin is introduced into the loss function, we find that $s$ will not increase and the network converges very slowly if we let $s$ to be learned. Thus, we fix $s$ to be a big value, e.g., 30, to accelerate the optimization.

During training, $m$ is linearly increased from 0 to the target margin value. This annealing strategy was proposed in [12] and has been empirically proven to be effective to avoid network divergence.

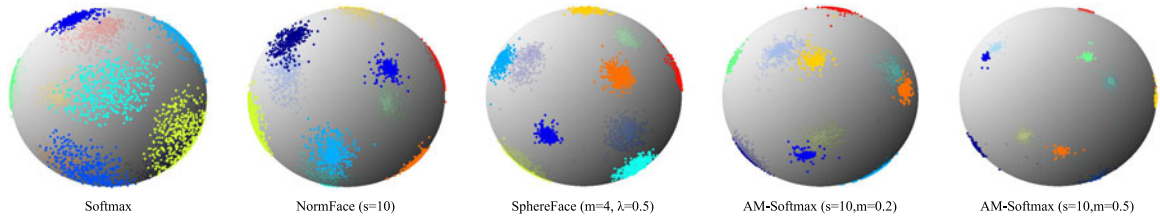


Fig. 4. Feature distribution visualization of several loss functions. Each point on the sphere represents one normalized feature. Different colors denote different classes. For SphereFace [11], the hyperparameter is almost the most harsh one we can set. When $\lambda$ is set below 0.5, the model starts to degrade and some of the classes are mixed up. The accuracies of these models are also provided. Note that the feature dimension is set to only 3 and those loss functions are not designed for classification except for softmax loss. The accuracies here are not comparable with state-of-the-art models.

### B. Discussion

*1) Geometric Explanation:* Our margin scheme has a clear geometric meaning on the hypersphere manifold. In Fig. 2, we draw a schematic diagram to show the decision boundary of both conventional softmax loss and our AM-Softmax. The decision boundary for softmax loss is a $d-2$ dimensional hyperplane on $d-1$ dimensional hypersphere in $d$-dimensional feature space. For example, in Fig. 2, the features are of two dimensions. After normalization, the features are on a one-dimensional (1-D) circle and the decision boundary $P_0$ is a 0-D point. In this condition, we have $W_1^T P_0 = W_2^T P_0$ at the boundary $P_0$.

For our AM-Softmax, the boundary becomes a marginal region. At the new boundary $P_1$ for class 1, we have $W_1^T P_1 - m = W_2^T P_1$. If we assume that all the classes have the same intra-class variance and the boundary for class 2 is at $P_2$, we can get $W_1^T P_1 - m = W_1^T P_2$. Thus, $m = W_1^T (P_1 - P_2)$, which means that the margin $m$ is actually the width of the marginal region.

*2) Angle Margin or Cosine Margin:* In SphereFace [11], the margin $m$ is multiplied on $\theta$, thus we call this kind of margin as *multiplicative margin*. In our proposed loss function, the margin is subtracted from $\cos\theta$. We call our margin form as *additive margin*. Someone may find that despite the operation, these two margin forms are different in the base values, one is $\theta$ and another is $\cos\theta$. So the question comes, whether to use the angle margin or the cosine margin?

Our answer is that it depends on which similarity measurement (or distance) the final loss function is optimizing. Obviously, our modified softmax loss function is optimizing the cosine similarity, not the angle. This may not be a problem if we are using the conventional softmax loss because the decision boundaries are the same in these two forms ($\cos\theta_1 = \cos\theta_2 \Rightarrow \theta_1 = \theta_2$). However, when we are trying to push the boundary, we will face a problem that these two similarities (distances) have different densities. Cosine values are more dense when the angles are near 0 or $\pi$.

If we want to optimize the angle, an $\arccos$ operation is necessary after $W^T \mathbf{f}$ is gotten.

*3) Feature Normalization:* In the SphereFace model [11], the authors added the weight normalization based on large margin softmax [12], leaving the feature still not normalized. Our loss function, same with [14], [17], [21], applies feature normalization and uses a global scale factor $s$ to replace the sample-specific feature norm in SphereFace [11]. So whether add the feature normalization or not?

TABLE I
EFFECT OF OVERLAP REMOVAL ON MODIFIED RESNET-20

| Loss Function | Overlap Removal? | MegaFace Rank1 | MegaFace VR |
|---|---|---|---|
| A-Softmax[10] | No | 73.69% | 85.05% |
| A-Softmax[10] | Yes | 67.41% | 78.19% |
| AM-Softmax | No | 75.23% | 87.06% |
| AM-Softmax | Yes | 72.47% | 84.44% |

Our answer is that it depends on the image quality. In [17, Fig. 1], we can see that the feature norm is highly correlated with the quality of the image. Note that backpropagation has a property that

$$y = \frac{x}{\alpha} \quad \Rightarrow \quad \frac{dy}{dx} = \frac{1}{\alpha}. \tag{7}$$

Thus, after normalization, features with small norms will get much bigger gradient compared with features that have big norms (see Fig. 3). By backpropagation, the network will pay more attention to the low-quality face images, which usually have small norms. Its effect is very similar with hard sample mining [10], [18]. The advantages of feature normalization are also revealed in [13]. As a conclusion, feature normalization is the best suitable for tasks whose image quality is very low.

From Fig. 3, we can see that the gradient norm may be extremely big when the feature norm is very small. This potentially increases the risk of gradient explosion, even though we may not come across many samples with very small feature norm. Maybe some re-weighting strategy whose feature-gradient norm curve is between the two curves in Fig. 3 could work better. This is an interesting topic to be studied in the future.

*4) Feature Distribution Visualization:* To better understand the effect of our loss function, we designed a toy experiment to visualize the feature distributions trained by several loss functions. We used Fashion MNIST [24] to train several seven-layer CNN models, which output 3-D features. The state-of-the-art results on this dataset are similar with Cifar-10 dataset [7], but it is much easier to train CNN models on it. After we got the 3-D features, we normalize and plot them on a two-sphere (ball) in 3-D space (see Fig. 4).

From the visualization, we can intuitively conclude that our AM-Softmax performs similarly with the best SphereFace [11] (A-Softmax) model when we set $s = 10$ and $m = 0.2$. Moreover, our loss function can further shrink the intraclass variance

TABLE II
PERFORMANCE ON MODIFIED RESNET-20 WITH VARIOUS LOSS FUNCTIONS

| Loss Function | $m$ | LFW [4] 6,000 pairs | LFW BLUFR [9] VR@FAR = 0.01% | LFW BLUFR [9] VR@FAR = 0.1% | LFW BLUFR [9] DIR@FAR = 1% | MegaFace [6] Rank1@1e6 | MegaFace [6] VR@FAR = 1e−6 |
|---|---|---|---|---|---|---|---|
| Softmax | - | 97.08% | 60.26% | 78.26% | 50.85% | 45.26% | 50.12% |
| Softmax+75% dropout | - | 98.62% | 77.64% | 90.91% | 63.72% | 57.32% | 65.58% |
| Center Loss [23] | - | 99.00% | 83.30% | 94.50% | 65.46% | 63.38% | 75.68% |
| NormFace [21] | - | 98.98% | 88.15% | 96.16% | 75.22% | 65.03% | 75.88% |
| A-Softmax [11] | ∼1.5 | 99.08% | 91.26% | 97.06% | 81.93% | 67.41% | 78.19% |
| AM-Softmax | 0.25 | 99.13% | 91.97% | 97.13% | 81.42% | 70.81% | 83.01% |
| AM-Softmax | 0.3 | 99.08% | 93.18% | 97.56% | 84.02% | 72.01% | 83.29% |
| AM-Softmax | 0.35 | 98.98% | 93.51% | 97.69% | 84.82% | **72.47%** | **84.44%** |
| AM-Softmax | 0.4 | 99.17% | 93.60% | 97.71% | 84.51% | 72.44% | 83.50% |
| AM-Softmax | 0.45 | 99.03% | 93.44% | 97.60% | 84.59% | 72.22% | 83.00% |
| AM-Softmax | 0.5 | 99.10% | 92.33% | 97.28% | 83.38% | 71.56% | 82.49% |
| AM-Softmax w/o FN | 0.35 | 99.08% | 93.86% | 97.63% | **87.58%** | 70.71% | 82.66% |
| AM-Softmax w/o FN | 0.4 | 99.12% | **94.48%** | **97.96%** | 87.31% | 70.96% | 83.11% |

*Note*: Forcenter loss [23] and NormFace [21], we used modified ResNet-28 [23] because we failed to train a model using center loss on modified ResNet-20 [11] and the NormFace model was fine-tuned based on the center loss model. In this table, "VR" is short for verification rate and "FN" is short for feature normalization. Bold values are the highest scores among the tested loss functions.

by setting a bigger $m$ without performance degradation, while A-Softmax starts to degrade when $\lambda < 0.5$.

## III. EXPERIMENT

### A. Implementation Details

Our loss function is implemented using Caffe framework [5]. We follow all the experimental settings from [11], including the image resolution, preprocessing method, and the network structure.

Specifically speaking, we use MTCNN [26] to detect faces and facial landmarks in images. Then, the faces are aligned according to the detected landmarks. The aligned face images are of size $112 \times 96$, and are normalized by subtracting 128 and dividing 128. Our network structure follows [11], which is a modified ResNet [2] with 20 layers.

All the networks are trained from scratch. We set the weight decay parameter to be 5e−4. The batch size is 256 and the learning rate begins with 0.1 and is divided by 10 at the 16k, 24k, and 28k iterations. The training is finished at 30k iterations. During training, we only use image mirror to augment the dataset.

In the testing phase, we feed both frontal face images and mirror face images and extract the features from the output of the first inner-product layer. Then, the two features are summed together as the representation of the face image. When comparing two face images, cosine similarity is utilized as the measurement.

### B. Dataset Overlap Removal

The dataset we use for training is CASIA-Webface [25], which contains 494 414 training images from 10 575 identities. To perform open-set evaluations, we carefully remove the overlapped identities between training dataset (CASIA-Webface [25]) and testing datasets (LFW [4] and MegaFace [6]). Finally, we find 17 overlapped identities between CASIA-Webface and LFW, and 42 overlapped identities between CASIA-Webface and MegaFace set1. Note that there are only 80 identities in MegaFace set1, i.e., over half of the identities are already in the training dataset. The effect of overlap removal is remarkable for MegaFace (see Table I). To be academically rigorous, all the experiments in this letter are based on the cleaned dataset. We have made our overlap checking code publicly available[1] to encourage researchers to clean their training datasets before experiments.

Since we know that most of previous works did not remove the duplicated identities sufficiently, we select some previous loss functions and re-train them on the cleaned dataset as the baselines for comparison.

### C. Effect of Hyperparameter $m$

There are two hyperparameters in our proposed loss function, one is the scale $s$ and another is the margin $m$. The scale $s$ has already been discussed sufficiently in several previous works [14], [17], [21]. In this letter, we directly fixed it to 30 and will not discuss its effect anymore.

The main hyperparameter in our loss function is the margin $m$. In Table II, we list the performance of our proposed AM-Softmax loss function with $m$ varies from 0.25 to 0.5. From the table, we can see that from $m = 0.25$ to 0.3, the performance improves significantly, and the performance become the best when $m = 0.35$–0.4.

We also provide the result for the loss function without feature normalization (noted as w/o FN) and the scale $s$. As we explained before, feature normalization performs better on low-quality images, such as MegaFace [6], and using the original feature norm performs better on high-quality images, such as LFW [4].

## IV. CONCLUSION

In this letter, we propose to import a margin strategy to the target logit of softmax loss with feature and weight normalized. We also provide a geometric explanation to analyze the decision boundary for both softmax and our AM-Softmax. Experiment shows that our loss function performs better than A-Softmax [11] on BLUFR [9] and MegaFace [6].

[1] https://github.com/happynear/FaceDatasets

## REFERENCES


[1] J. Deng, J. Guo, and S. Zafeiriou, "ArcFace: Additive angular margin loss for deep face recognition," arXiv:1801.07698, 2018.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[3] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," arXiv:1503.02531, 2015.

[4] 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," Univ. of Massachusetts, Amherst, MA, USA, Tech. Rep. 07-49, 2007.

[5] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. 22nd ACM Int. Conf. Multimedia*, 2014, pp. 675–678.

[6] I. Kemelmacher-Shlizerman, S. M. Seitz, D. Miller, and E. Brossard, "The MegaFace benchmark: 1 million faces for recognition at scale," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 4873–4882.

[7] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Master thesis, Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009.

[8] X. Liang, X. Wang, Z. Lei, S. Liao, and S. Z. Li, "Soft-margin softmax for deep classification," in *Proc. 24th Int. Conf. Neural Inf. Process.*, 2017, pp. 413–421.

[9] S. Liao, Z. Lei, D. Yi, and S. Z. Li, "A benchmark study of large-scale unconstrained face recognition," in *Proc. IEEE Int. Joint Conf. Biometrics*, 2014, pp. 1–8.

[10] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2999–3007.

[11] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "SphereFace: Deep hypersphere embedding for face recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6738–6746.

[12] W. Liu, Y. Wen, Z. Yu, and M. Yang, "Large-margin softmax loss for convolutional neural networks," in *Proc. 33rd Int. Conf. Mach. Learn.*, 2016, pp. 507–516.

[13] W. Liu *et al.*, "Deep hyperspherical learning," in *Proc. Advances Neural Inf. Process. Syst.*, 2017, pp. 3953–3963.

[14] Y. Liu, H. Li, and X. Wang, "Rethinking feature discrimination and polymerization for large-scale recognition," in *Proc. Advances Neural Inf. Process. Syst. Workshop*, arXiv:1710.00870, 2017.

[15] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *Proc. Br. Mach. Vis. Conf.*, 2015, vol. 1, pp. 41.1–41.12.

[16] G. Pereyra, G. Tucker, J. Chorowski, Ł. Kaiser, and G. Hinton, "Regularizing neural networks by penalizing confident output distributions," in *Proc. Int. Conf. Learn. Represent. Workshop*, arXiv:1701.06548, 2017.

[17] R. Ranjan, C. D. Castillo, and R. Chellappa, "L2-constrained softmax loss for discriminative face verification," arXiv:1703.09507, 2017.

[18] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 815–823.

[19] Y. Sun, Y. Chen, X. Wang, and X. Tang, "Deep learning face representation by joint identification-verification," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 1988–1996.

[20] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: Closing the gap to human-level performance in face verification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 1701–1708.

[21] F. Wang, X. Xiang, J. Cheng, and A. L. Yuille, "NormFace: L2 hypersphere embedding for face verification," in *Proc. 25th ACM Int. Conf. Multimedia*, 2017, pp. 1041–1049.

[22] H. Wang *et al.*, "CosFace: Large margin cosine loss for deep face recognition," arXiv:1801.09414, 2018.

[23] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 499–515. .

[24] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017.

[25] D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Learning face representation from scratch," arXiv:1411.7923, 2014.

[26] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Process. Lett.*, vol. 23, no. 10, pp. 1499–1503, Oct. 2016.