ADDITIVE MARGIN SOFTMAX FOR FACE VERIFICATION

Feng Wang, Haijun Liu, Jian Cheng
Department of Information and Communication Engineering
University of Electronic Science and Technology of China
Chengdu, Sichuan 611731 China
feng.wff@gmail.com haijun_liu@126.com chengjian@uestc.edu.cn

Weiyang Liu
College of Computing
Georgia Institute of Technology
Atlanta, United States.
wyliu@gatech.edu

ABSTRACT

In this paper, we propose a conceptually simple and geometrically interpretable objective function, i.e. additive margin Softmax (AM-Softmax), for deep face verification. In general, the face verification task can be viewed as a metric learning problem, so learning large-margin face features whose intra-class variation is small and inter-class difference is large is of great importance in order to achieve good performance. Recently, Large-margin Softmax Liu et al. (2016) and Angular Softmax Liu et al. (2017a) have been proposed to incorporate the angular margin in a multiplicative manner. In this work, we introduce a novel additive angular margin for the Softmax loss, which is intuitively appealing and more interpretable than the existing works. We also emphasize and discuss the importance of feature normalization in the paper. Most importantly, our experiments on LFW and MegaFace show that our additive margin softmax loss consistently performs better than the current state-of-the-art methods using the same network architecture and training dataset. Our code has also been made available[1].

1 INTRODUCTION

Face verification is widely used for identity authentication in enormous areas such as finance, military, public security and so on. Nowadays, most face verification models are built upon Deep Convolutional Neural Networks and supervised by classification loss functions Taigman et al. (2014); Wen et al. (2016); Wang et al. (2017); Liu et al. (2017a), metric learning loss functions Schroff et al. (2015) or both Sun et al. (2014); Parkhi et al. (2015). Metric learning loss functions such as contrastive loss Sun et al. (2014) or triplet loss Schroff et al. (2015) usually require carefully designed sample mining strategies and the final performance is very sensitive to these strategies, so increasingly more researchers shift their attentions to building deep face verification models based on improved classification loss functions Wen et al. (2016); Wang et al. (2017); Liu et al. (2017a).

Our work is based on two of these loss functions, NormFace Wang et al. (2017)

$$\mathcal{L}_{NS} = -\frac{1}{n}\sum_{i=1}^{n} log\frac{e^{s\cdot cos\theta_{y_i}}}{\sum_{j=1,j}^{c} e^{s\cdot cos\theta_j}}, \tag{1}$$

and SphereFace Liu et al. (2017a)

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

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

where the $\psi(\theta)$ is defined as

$$\psi(\theta) = \frac{(-1)^k \cos(m\theta) - 2k + \lambda cos(\theta)}{1 + \lambda}, \theta \in [\frac{k\pi}{m}, \frac{(k+1)\pi}{m}]. \tag{3}$$

In these two loss functions, $\boldsymbol{f}$ is the input of the last fully connected layer ($\boldsymbol{f}_i$ denotes the the i-th sample), W_j is the j-th column of the last fully connected layer. The $W_{y_i}^T \boldsymbol{f}_i$ is also called as the target logit Pereyra et al. (2017) of the i-th sample. The s is a scale parameter that enables the Softmax loss converge (Proposition 2 in Wang et al. (2017)). The m is an integer larger than 1 to create a margin area with angle span of $(m-1)\theta$. The λ is a balance parameter to make the model easier to converge.

In this work, 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 Liu et al. (2017a) and yields better performance. From Equation (8), we can see that m is multiplied to the target angle θ_{y_i} in Liu et al. (2017a), so this type of margin is incorporated in a multiplicative manner. Since our margin is a scalar subtracted from $cos\theta$, we call our loss function Additive Margin Softmax (AM-Softmax).

Experiments on LFW BLUFR protocol Liao et al. (2014) and MegaFace Kemelmacher-Shlizerman et al. (2016) show that our loss function with the same network architecture achieves better results than the current state-of-the-art approaches.

2 Additive Margin Softmax

Motivated by Equation (8), we further propose a specific $\psi(\theta)$ that introduces an additive margin to the softmax loss function. The formulation is given by

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

Compared to the $\psi(\theta)$ defined in L-Softmax Liu et al. (2016) and A-softmax Liu et al. (2017a) (Equation (8)), our definition is more simple and intuitive. During implementation, the input after normalizing both the feature and the weight is actually $x = cos\theta_{y_i} = \frac{W_{y_i}^T f_i}{\|W_{y_i}\| \|f_i\|}$, so in the forward propagation we only need to compute

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

In this margin scheme, we don't need to calculate the gradient for back-propagation because $\Psi'(x) = 1$. It is much easier to implement compared with SphereFace Liu et al. (2017a).

Since we use cosine as the similarity to compare two face features, we follow Wang et al. (2017) 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 hyper-parameter s as suggested in Wang et al. (2017); Liu et al. (2017b;c). Finally, the loss function becomes

$$\mathcal{L}_{AMS} = -\frac{1}{n} \sum_{i=1}^{n} log \frac{e^{s \cdot (cos\theta_{y_i} - m)}}{e^{s \cdot (cos\theta_{y_i} - m)} + \sum_{j=1, j \neq y_i}^{c} e^{s \cdot cos\theta_j}} \tag{6}$$

In Wang et al. (2017), the authors propose to let the scaling factor s to be learned through back-propagation. However, after the margin is introduced into the loss function, we find that the s will not increase and the network converges very slowly if we let s to be learned. Thus, we fix s to be a large enough value, e.g. 30, to accelerate and stablize the optimization.

3 Experiments

3.1 Implementation Details

Our loss function is implemented using Caffe framework Jia et al. (2014). We follow all the experimental settings from Liu et al. (2017a), including the image resolution, preprocessing method and the network structure. Specifically speaking, we use MTCNN Zhang et al. (2016) to detect faces

Loss Function	m	LFW BLUFR VR@FAR=0.01%	LFW BLUFR DIR@FAR=1%	MegaFace Rank1@1e6	MegaFace VR@FAR=1e$-$6
Softmax	-	60.26%	50.85%	45.26%	50.12%
Softmax+75% dropout	-	77.64%	63.72%	57.32%	65.58%
Wen et al. (2016)	-	83.30%	65.46%	63.38%	75.68%
Wang et al. (2017)	-	88.15%	75.22%	65.03%	75.88%
Liu et al. (2017a)	$\sim$1.5	91.26%	81.93%	67.41%	78.19%
AM-Softmax	0.25	91.97%	81.42%	70.81%	83.01%
AM-Softmax	0.3	93.18%	84.02%	72.01%	83.29%
AM-Softmax	0.35	93.51%	84.82%	**72.47%**	**84.44%**
AM-Softmax	0.4	93.60%	84.51%	72.44%	83.50%
AM-Softmax	0.45	93.44%	84.59%	72.22%	83.00%
AM-Softmax	0.5	92.33%	83.38%	71.56%	82.49%
AM-Softmax w/o FN	0.35	93.86%	**87.58%**	70.71%	82.66%
AM-Softmax w/o FN	0.4	**94.48%**	87.31%	70.96%	83.11%

Table 1: Performance on modified ResNet-20 with various loss functions. Note that, for Center Loss Wen et al. (2016) and NormFace Wang et al. (2017), we used modified ResNet-28 Wen et al. (2016) because we failed to train a model using Center Loss on modified ResNet-20 Liu et al. (2017a) and the NormFace model was fine-tuned based on the Center Loss model.

and facial landmarks in images. Then the faces are aligned according to the detected landmarks. The aligned face images are of size 112×96, and are normalized by subtracting 128 and dividing 128. Our network structure follows Liu et al. (2017a), which is a modified ResNet He et al. (2016) with 20 layers that is adapted to face recognition.

All the networks are trained from scratch. We set the weight decay parameter to be $5\text{e}{-4}$. 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. The dataset we use for training is CASIA-Webface Yi et al. (2014), 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 Yi et al. (2014)) and testing datasets (LFWHuang et al. (2007) and MegaFace Kemelmacher-Shlizerman et al. (2016)). In 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.

3.2 Effect of Hyper-parameter m

There are two hyper-parameters 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 Wang et al. (2017); Liu et al. (2017c); Ranjan et al. (2017). In this paper, we directly fixed it to 30 and will not discuss its effect anymore.

The main hyper-parameter in our loss function is the margin m. In Table 3.1, 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 $m = 0.35$ to $m = 0.4$.

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 like MegaFaceKemelmacher-Shlizerman et al. (2016), and using the original feature norm performs better on high quality images like LFW Huang et al. (2007).

REFERENCES

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

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.

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pp. 675–678. ACM, 2014.

Ira Kemelmacher-Shlizerman, Steven M Seitz, Daniel Miller, and Evan Brossard. The megaface benchmark: 1 million faces for recognition at scale. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4873–4882, 2016.

Shengcai Liao, Zhen Lei, Dong Yi, and Stan Z Li. A benchmark study of large-scale unconstrained face recognition. In *IEEE International Joint Conference on Biometrics*, pp. 1–8. IEEE, 2014.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *arXiv preprint arXiv:1708.02002*, 2017.

Weiyang Liu, Yandong Wen, Zhiding Yu, and Meng Yang. Large-margin softmax loss for convolutional neural networks. In *International Conference on Machine Learning*, pp. 507–516, 2016.

Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017a.

Weiyang Liu, Yan-Ming Zhang, Xingguo Li, Zhiding Yu, Bo Dai, Tuo Zhao, and Le Song. Deep hyperspherical learning. In *Advances in Neural Information Processing Systems*, pp. 3953–3963, 2017b.

Yu Liu, Hongyang Li, and Xiaogang Wang. Rethinking feature discrimination and polymerization for large-scale recognition. *arXiv preprint arXiv:1710.00870*, 2017c.

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

Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, 2017.

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

Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 815–823, 2015.

Yi Sun, Yuheng Chen, Xiaogang Wang, and Xiaoou Tang. Deep learning face representation by joint identification-verification. In *Advances in neural information processing systems*, pp. 1988–1996, 2014.

Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1701–1708, 2014.

Feng Wang, Xiang Xiang, Jian Cheng, and Alan L. Yuille. Normface: L2 hypersphere embedding for face verification. In *Proceedings of the 25th ACM international conference on Multimedia*. ACM, 2017. doi: https://doi.org/10.1145/3123266.3123359.

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

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

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

Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, 2016.

4 APPENDIX

4.1 THE EFFECT OF m AND λ IN SPHEREFACE LIU ET AL. (2017A)

The authors of SphereFace Liu et al. (2017a) proposed to normalize the weight vectors (making $\|W_i\|$ to be 1) and generalize the target logit from $\|\boldsymbol{f}_i\|cos(\theta_{y_i})$ to $\|\boldsymbol{f}_i\|\psi(\theta_{y_i})$,

$$\mathcal{L}_{AS} = -\frac{1}{n}\sum_{i=1}^{n} log\frac{e^{\|\boldsymbol{f}_i\|\psi(\theta_{y_i})}}{e^{\|\boldsymbol{f}_i\|\psi(\theta_{y_i})} + \sum_{j=1,j\neq y_i}^{c} e^{\|\boldsymbol{f}_i\|cos(\theta_j)}}, \tag{7}$$

where the $\psi(\theta)$ is usually a piece-wise function defined as

$$\psi(\theta) = \frac{(-1)^k \cos(m\theta) - 2k + \lambda cos(\theta)}{1+\lambda}, \quad \theta \in [\frac{k\pi}{m}, \frac{(k+1)\pi}{m}], \tag{8}$$

where m is usually an integer larger than 1 and λ is a hyper-parameter to control how hard the classification boundary should be pushed. During training, the λ is annealing from $1,000$ to a small value to make the angular space of each class become more and more compact. In their experiments, they set the minimum value of λ to be 5 and $m = 4$, which is approximately equivalent to $m = 1.5$ (Figure 1).

We also draw our modified $\psi(\theta)$ in Figure 1. From the figure, it can be observed that our version of $\psi(\theta)$ with the optimized parameter $m = 0.35$ is similar with SphereFace's curve with $m = 4, \lambda = 5$ in the range $[0°, 90°]$, in which most of the real-world θs lie.

4.2 GEOMETRIC INTERPRETATION

Our additive margin scheme has a clear geometric interpretation on the hypersphere manifold. In Figure 2, we draw a schematic diagram to show the decision boundary of both conventional softmax loss and our AM-Softmax. For example, in Figure 2, the features are of 2 dimensions. After normalization, the features are on a circle and the decision boundary of the traditional softmax loss is denoted as the vector P_0. In this case, we have $W_1^T P_0 = W_2^T P_0$ at the decision boundary P_0.

For our AM-Softmax, the boundary becomes a marginal region instead of a single vector. At the new boundary P_1 for class 1, we have $W_1^T P_1 - m = W_2^T P_1$, which gives $m = (W_1 - W_2)^T P_1 = \cos(\theta_{W_1,P_1}) - \cos(\theta_{W_2,P_1})$. If we further assume that all the classes have the same intra-class variance and the boundary for class 2 is at P_2, we can get $\cos(\theta_{W_2,P_1}) = \cos(\theta_{W_1,P_2})$ (Fig. 2). Thus, $m = \cos(\theta_{W_1,P_1}) - \cos(\theta_{W_1,P_2})$, which is the difference of the cosine scores for class 1 between the two sides of the margin region.

4.3 ANGULAR MARGIN OR COSINE MARGIN

In SphereFace Liu et al. (2017a), the margin m is multiplied to θ, so the angular margin is incorporated into the loss in a multiplicative way. In our proposed loss function, the margin is enforced by

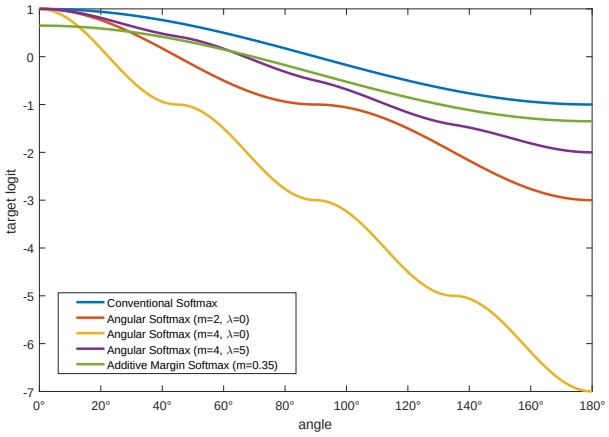

Figure 1: $\psi(\theta)$ for conventional Softmax, Angular Softmax Liu et al. (2017a) and our proposed Hard Margin Softmax. For Angular Softmax, we plot the logit curve for three parameter sets. From the curves we can infer that $m = 4, \lambda = 5$ lies between conventional Softmax and Angular Softmax with $m = 2, \lambda = 0$, which means it is approximately $m = 1.5$. Our proposed Additive Margin Softmax with optimized parameter $m = 0.35$ is also plotted and we can observe that it is similar with Angular Softmax with $m = 4, \lambda = 5$ in the range $[0°, 90°]$, in which most of the real-world θs lie.

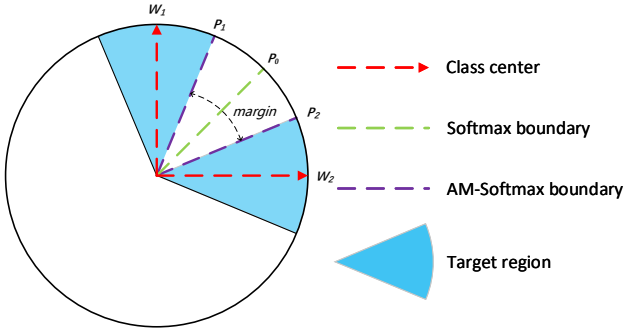

Figure 2: Conventional Softmax's decision boundary and Additive Margin Softmax's decision boundary. For conventional softmax, the decision boundary is at P_0, where $W_1^T P_0 = W_2^T P_0$. For AM-Softmax, the decision boundary for class 1 is at P_1, where $W_1^T P_1 - m = W_2^T P_1 = W_1^T P_2$. Note that the distance marked on this figure doesn't represent the real distances. The real distance is a function of the cosine of the angle, while 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 Figure 6 of Wang et al. (2017).

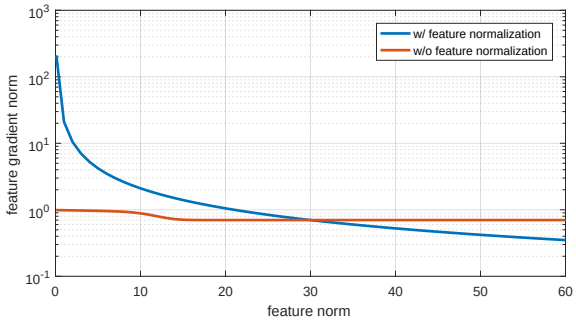

Figure 3: The feature gradient norm w.r.t. the feature norm for softmax loss with and without feature normalization. The gradients are calculated using the weights from a converged network. The feature direction is selected as the mean vector of one selected target center and one nearest class center. 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.

subtracting m from $\cos\theta$, so our margin is incorporated into the loss in an additive way, which is one of the most significant differences than Liu et al. (2017a). It is also worth mentioning that despite the difference of enforcing margin, these two types of margin formulations are also different in the base values. Specifically, one is θ and the other is $\cos\theta$. Although usually the cosine margin has an one-to-one mapping to the angular margin, there will still be some difference while optimizing them due to the non-linearity induced by the cosine function.

Whether we should use the cosine margin 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 π. If we want to optimize the angle, an $\arccos$ operation may be required after the value of the inner product $W^T \boldsymbol{f}$ is obtained. It will potentially be more computationally expensive.

In general, angular margin is conceptually better than the cosine margin, but considering the computational cost, cosine margin is more appealing in the sense that it could achieve the same goal with less efforts.

4.4 Feature Normalization

In the SphereFace model Liu et al. (2017a), the authors added the weight normalization based on Large Margin Softmax Liu et al. (2016), leaving the feature still not normalized. Our loss function, following Wang et al. (2017); Liu et al. (2017c); Ranjan et al. (2017), applies feature normalization and uses a global scale factor s to replace the sample-dependent feature norm in SphereFace Liu et al. (2017a). One question arises: when should we add the feature normalization?

Our answer is that it depends on the image quality. In Ranjan et al. (2017)'s Figure 1, we can see that the feature norm is highly correlated with the quality of the image. Note that back propagation has a property that,

$$y = \frac{x}{\alpha} \quad \Rightarrow \quad \frac{dy}{dx} = \frac{1}{\alpha}. \tag{9}$$

Thus, after normalization, features with small norms will get much bigger gradient compared with features that have big norms (Figure 3). By back-propagation, 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 Schroff et al. (2015); Lin et al. (2017). The advantages of feature normalization are also revealed in Liu et al. (2017b). As a conclusion, feature normalization is most suitable for tasks whose image quality is very low.

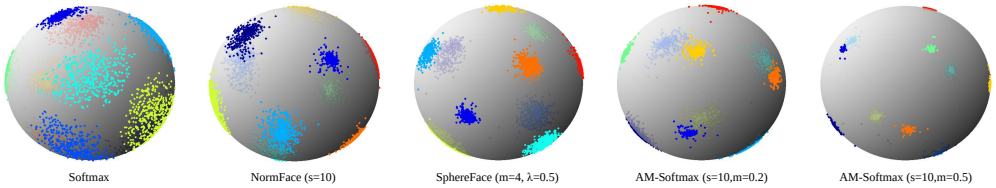

Figure 4: Feature distribution visualization of several loss functions. Each point on the sphere represent one normalized feature. Different colors denote different classes. For SphereFace Liu et al. (2017a), we have already tried to use the best hyper-parameters we could find.

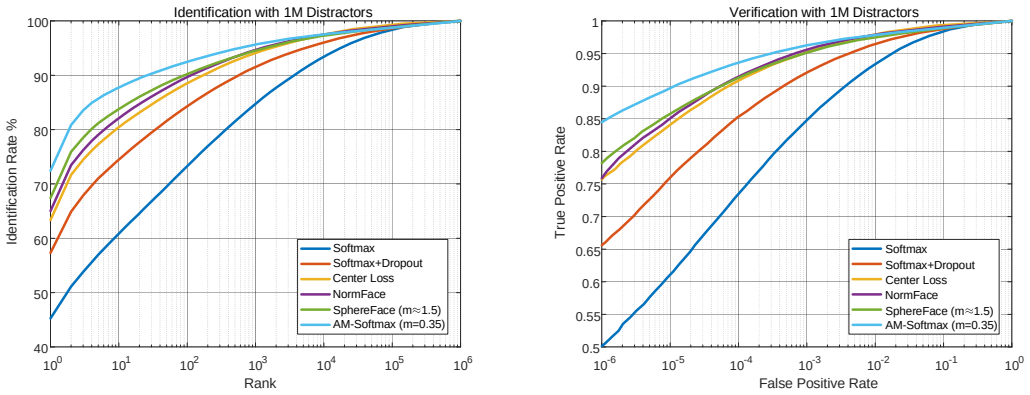

Figure 5: *Left:* CMC curves of different loss functions with 1M distractors on MegaFaceKemelmacher-Shlizerman et al. (2016) Set 1. *Right:* ROC curves of different loss functions with 1M distractors on MegaFaceKemelmacher-Shlizerman et al. (2016) Set 1. Note that for Center Loss and NormFace, the backend network is ResNet-28Wen et al. (2016), while others are based on ResNet-20Liu et al. (2017a). Even though the curves of the Center Loss model and the NormFace model is close to the SphereFace model, please keep in mind that part of the performance comes from the bigger network structure.

From Figure 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 Figure 3 could potentially work better. This is an interesting topic to be studied in the future.

4.5 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 Xiao et al. (2017) (10 classes) to train several 7-layer CNN models which output 3-dimensional features. These networks are supervised by different loss functions. After we obtain the 3-dimensional features, we normalize and plot them on a hypersphere (ball) in the 3 dimensional space (Figure 4).

From the visualization, we can empirically show that our AM-Softmax performs similarly with the best SphereFace Liu et al. (2017a) (A-Softmax) model when we set $s = 10, m = 0.2$. Moreover, our loss function can further shrink the intra-class variance by setting a larger m. Compared to A-Softmax Liu et al. (2017a), the AM-Softmax loss also converges easier with proper scaling factor s. The visualized 3D features well demonstrates that AM-Softmax could bring the large margin property to the features without tuning too many hyper-parameters.

4.6 ROC AND CMC CURVES ON MEGAFACE

In Figure 5, we draw both of the CMC curves to evaluate the performance of identification and ROC curves to evaluate the performance of verification on MegaFace datasetKemelmacher-Shlizerman et al. (2016). From this figure, we can show that our loss function performs much better than the other loss functions when the rank or false positive rate is very low.