AdaptiveFace: Adaptive Margin and Sampling for Face Recognition

Hao Liu[1,2] Xiangyu Zhu[1,2] Zhen Lei[1,2*] Stan Z. Li[1,2,3]
[1]CBSR & NLPR, Institute of Automation, Chinese Academy of Sciences, Beijing, China.
[2]University of Chinese Academy of Sciences, Beijing, China.
[3]Faculty of Information Technology, Macau University of Science and Technology, Macau SAR, China.
{hao.liu2016,xiangyu.zhu,zlei,szli}@nlpr.ia.ac.cn

Abstract

Training large-scale unbalanced data is the central topic in face recognition. In the past two years, face recognition has achieved remarkable improvements due to the introduction of margin based Softmax loss. However, these methods have an implicit assumption that all the classes possess sufficient samples to describe its distribution, so that a manually set margin is enough to equally squeeze each intra-class variations. However, real face datasets are highly unbalanced, which means the classes have tremendously different numbers of samples. In this paper, we argue that the margin should be adapted to different classes. We propose the Adaptive Margin Softmax to adjust the margins for different classes adaptively. In addition to the unbalance challenge, face data always consists of large-scale classes and samples. Smartly selecting valuable classes and samples to participate in the training makes the training more effective and efficient. To this end, we also make the sampling process adaptive in two folds: Firstly, we propose the Hard Prototype Mining to adaptively select a small number of hard classes to participate in classification. Secondly, for data sampling, we introduce the Adaptive Data Sampling to find valuable samples for training adaptively. We combine these three parts together as AdaptiveFace. Extensive analysis and experiments on LFW, LFW BLUFR and MegaFace show that our method performs better than state-of-the-art methods using the same network architecture and training dataset. Code is available at `https://github.com/haoliu1994/AdaptiveFace`.

1. Introduction

Face recognition, as one of the most common computer vision tasks, has made dramatic improvements in recent years [8, 3, 24, 36, 44, 33, 23, 18, 45, 40, 19, 5, 2]. It is worth noting that in the last few years, most approaches focus on loss functions which aim to reduce intra-class variations and enlarge inter-class variations. As one of milestone contributions, the margin based Softmax [17, 16, 34, 32, 4] explicitly adds a margin to each identity to improve feature discrimination. For example, L-Softmax [17] and SphereFace [16] add multiplicative angular margin to squeeze each class. CosFace [34, 32] and ArcFace [4] achieve the state-of-the-art performance by adding additive cosine margin and angular margin respectively for easier optimization. Nonetheless, these methods have an implicit hypothesis that all the classes have sufficient samples to describe their distributions, so that a constant margin is enough to equally squeeze each intra-class variations. However, public face datasets are highly unbalanced, which indicates that they always have tremendously different numbers of samples as shown in Fig 1. For those rich classes with sufficient samples, the space spanned by existing training samples can represent the real distribution. However, for those poor classes with scarce samples, the space spanned by the existing training samples may be only a small part of the real distribution. Therefore, a uniform margin is not perfect to constrain classes with different sample distributions. We prefer a larger margin to strongly squeeze the intra-class variations of those underrepresented classes to improve generalization. In this paper, we propose a novel loss function, Adaptive Margin Softmax Loss (AdaM-Softmax), to adaptively find the appropriate margins for different kind of classes. Specifically, we make the margin m particular and learnable for each class and directly train the CNN to find the adaptive margins. Formally, we define the margin of each class m_i such that the decision boundary is given by $\cos\theta_1 - m_1 = \cos\theta_2$, where θ_i is the angle between the feature and weight of class i. In the experiments, we show that AdaM-Softmax is superior to the baseline methods.

Besides, the large-scale face data always contains hundreds of thousands of classes and millions of samples, where only a small fraction of them can contribute to the discriminative training. How to select valuable classes and samples for training is another important topic but attracts

*Corresponding author

978-1-7281-3293-8/19/$31.00 ©2019 IEEE
DOI 10.1109/CVPR.2019.01222

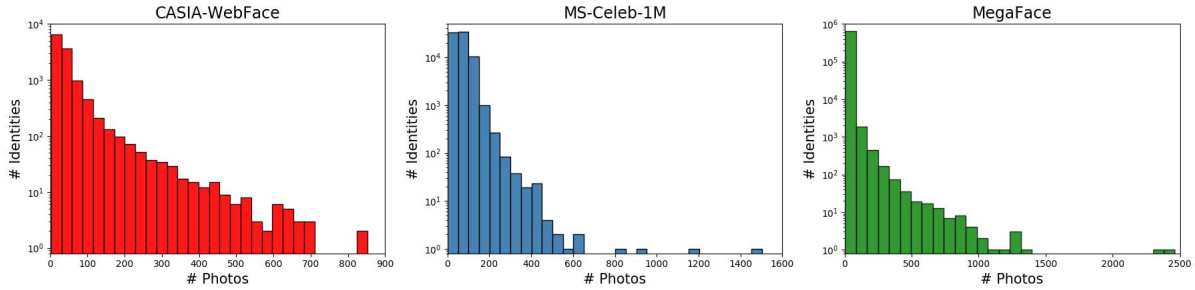

Figure 1. The distribution of photos per identity in CASIA-WebFace, MS-Celeb-1M and MegaFace datasets. We can see that the distributions of the three datasets are extremely unbalanced.

little attention in Softmax loss. In this paper, we also make the sampling process adaptive. The sampling refers to the prototype selection in Softmax layer and the data sampling in the data layer. Firstly, in deep metric learning, hard example mining is an important part to improve the training efficiency and performance of the model. Zhu *et al*. [47] show that both verification loss and classification loss follow the same pair matching and weighting framework. The only differences lie in the pairing candidates (within features vs. features with prototypes) and the weighting methods (hard weight vs. soft weight). Therefore, in this paper, we try to apply the hard example mining strategy in the Softmax loss. Specifically, we propose the Hard Prototype Mining (HPM) to adaptively select a small number of hard classes to participate in the classification, making the optimization concentrate on hard classes. Note that we regard the weight vector of each class as its prototype in this paper. Secondly, learning from large-scale data is crucial for current face recognition task, so that training efficiency becomes more and more important due to the limitation of time and computing devices. Inspired by mini-batch level hard example mining, we propose the Adaptive Data Sampling (ADS), which finds the valuable samples for network training through a feedback channel from the classification layer to the data layer. Based on the three components, we call the proposed face recognition framework AdaptiveFace, as shown in Figure 2.

In summary, we aim to make the face recognition framework more flexible to handle the large scale and unbalanced data. Our major contributions are as follows:

(1) We introduce the adaptive margin to make the model learn the special margin for each class to adaptively squeeze its intra-class variations.

(2) We propose the Hard Prototype Mining to make the network training concentrate on hard classes by adaptively mining a small number of hard prototypes during classification training.

(3) We establish a feedback channel from the classification layer to the data layer to find the valuable samples for network training.

Experiments on LFW, LFW BLUFR and MegaFace show that our method effectively improves the recognition accuracy and achieves state-of-the-art performance.

2. Related Works

In this section, we review the deep learning based face recognition and discuss two related problems: (1) Loss functions and (2) Hard example mining.

Loss Functions. Loss function plays an important role for face recognition. We will introduce the loss function from two aspects. The first is the verification loss function. Contrastive loss [3, 6, 27] optimizes pairwise Euclidean distance in feature space. Triplet loss [24, 9, 35] making up triplets to separate the positive pair from the negative pair by a distance margin. The second is the classification loss function. The most popular loss in this scheme is the Softmax loss [28, 30, 31]. Based on that, the center loss [36] proposes to learn the class-specific feature centers to make features more compact in the embedding space. The L2-softmax [23] adds a L2-constraint on features to promote the under-represented classes. The NormFace [33] normalizes both features and prototypes to make the training and testing phases closer. Recently, enhancing cosine and angular margins between different classes is found to be effective in improving feature discrimination. The large-margin Softmax [17] and A-Softmax [16] add multiplicative angular margin to each identity to improve feature discrimination. CosFace [34] and AM-Softmax [32] add additive cosine margin for better optimization. ArcFace [4] moves the additive cosine margin into angular space to get clear geometrical interpretation and better performance on a series of face recognition benchmarks.

Hard Example Mining. Hard example mining is an important part for deep metric learning to improve the training efficiency and performance of the model. The way to find hard examples is usually to use online hard example mining (OHEM) [10, 39]. However, in practice, because of amounts of noise in large scale data, it is better to use online semi-hard example mining [24, 21, 22], in which example pairs are chosen at random from the "hard enough" pairs in

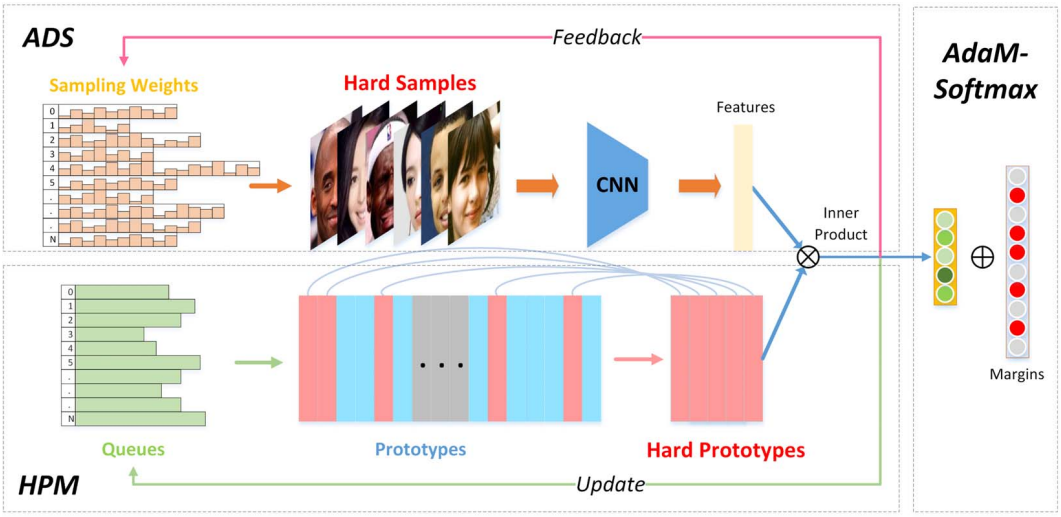

Figure 2. Overview of our AdaptiveFace. It consists of ADS, HPM and AdaM-Softmax.

the mini-batch. Also, not only hard pairs contain useful information [46], utilization of different levels of "hardness" has also proved to be beneficial [37, 41, 7]. All these methods improve hard sample mining, and there are others for mining hard classes. N-pair loss [26] uses "hard class mining" to find pairs of classes to generate mini-batches. Doppelganger Mining [25] maintains a list with the most similar identities for each identity to generate better mini-batches.

3. The Proposed Approach

In this section, we detail our methodology. In Section 3.1, we discuss why an identical margin cannot work well for classes with different numbers of samples and introduce our Adaptive Margin Softmax to find the special and appropriate margin for each class in an end-to-end way. In Section 3.2, we then propose the hard prototype mining to smartly select hard classes in the Softmax loss. Finally, we introduce a feedback channel to find the valuable samples for data sampling in Section 3.3.

3.1. Adaptive Margin Softmax

3.1.1 Intuition and Motivation

The recent works on margin based Softmax loss [17, 16, 34, 32, 4] have achieved significant improvements, where a manually tuned m is set for all the classes to squeeze the intra-class variations. There is an implicit assumption in these methods that the sample distributions of all the classes are identical, so that a manually set margin is enough to constrain all the classes. However, there is serious sample imbalance in existing face training data, as shown in Figure 1. For those classes with rich samples and large intra-class variations, the space spanned by existing training samples can represent the real distribution of all their samples, but for those poor classes with scarce samples and small intra-class variations, the space spanned by existing samples may be only a small part of the real distribution of this class. Note that, those classes with continuous frame of a tracklet are still considered as poor classes, since these frames provide few intra-class information. When a uniform margin is set for all the classes, the feature distributions of poor classes may not be as compact as that of rich classes, since the real spanned space of poor classes may be larger than the observed space, resulting in poor generalization.

We further visualize the phenomenon through a binary classification task as shown in Figure 3 (a). The blue area indicates the feature space of a poor class C_1 with scarce samples and small intra-class variations. The translucent blue area indicates the underlying real feature space of C_1, which cannot be observed due to the scarce samples. The red area indicates the feature space of class C_2 with rich samples and large intra-class variations. Since there are rich samples in C_2, we think observed feature space is almost the same as the underlying real feature space, so the translucent red area is the same as the red one. It can be seen that the CosFace loss cannot well compact the features of C_1 since it cannot see the real boundary samples. As a result, the underlying real feature space of C_1 can't be as compact as that of C_2. To address this issue, we propose the Adaptive Margin Softmax Loss (AdaM-Softmax), which improves the fixed margin m to a learnable and class-related parameter.

3.1.2 Adaptive Margin Softmax Loss

Let's start with the most widely used Softmax loss. The Softmax loss separates features from different classes by maximizing posterior probability of the ground-truth class. Given an input feature vector $\mathbf{x}_j$ with its corresponding la-

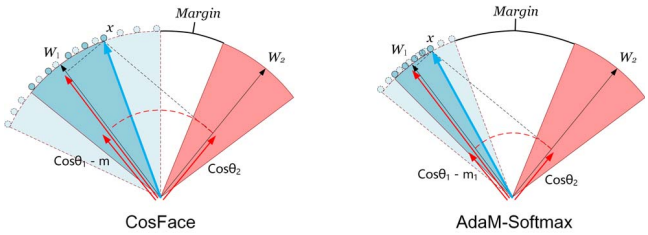

Figure 3. Geometrical interpretation of AdaM-Softmax from feature perspective. Blue area represents the feature space and translucent blue area is the underlying real feature space of the poor class C_1, red area represents the feature space of the rich class C_2. (a) CosFace allocates an identical margin for both classes, so that the poor class cannot be well compacted since the real boundary samples are not observed. (b) AdaM-Softmax allocates a larger margin to further compact the poor class, which implicitly optimize the underlying real space.

bel $y^{(j)}$, the Softmax loss without bias can be formulated as:

$$L_{soft} = -\frac{1}{M}\sum_{j=1}^{M}\log p_{ij} = -\frac{1}{M}\sum_{j=1}^{M}\log\left(\frac{e^{\mathbf{w}_{y^{(j)}}^{T}\mathbf{x}_j}}{\sum_{i=1}^{N} e^{\mathbf{w}_i^{\mathrm{T}}\mathbf{x}_j}}\right) \tag{1}$$

where p_{ij} denotes the posterior probability of $\mathbf{x}_j$ being correctly classified into class $y^{(j)}$, M is the batch size, N is the number of classes and $\mathbf{w}_i$ denotes the prototype of class i. Applying L2 normalization on $\mathbf{w}_i$ and $\mathbf{x}_j$ to optimize the feature on a sphere, the feature distances can be formulated as feature angulars as follows:

$$\mathbf{w}_i^{\mathrm{T}}\mathbf{x}_j = \|\mathbf{w}_i\|\|\mathbf{x}_j\|\cos\theta_{ij} = \cos\theta_{ij} \tag{2}$$

where θ_{ij} is the angle between $\mathbf{w}_i$ and $\mathbf{x}_j$. Based on this formulation, some methods introduce a margin to improve the intra-class compactness, such as A-Softmax [16], CosFace [34, 32] and ArcFace [4]. Take CosFace as an example:

$$L_{lmc} = -\frac{1}{M}\sum_{j=1}^{M}\log\frac{e^{s(\cos(\theta_{y^{(j)}j})-m)}}{e^{s(\cos(\theta_{y^{(j)}j})-m)}+\sum_{i=1,i\neq y^{(j)}}^{N} e^{s\cos(\theta_{ij})}} \tag{3}$$

where s is the scale factor. The margin m in CosFace are usually manually set and kept constant along the training process. In order to deal with the problem described in the section 3.1.1, we aim to improve the margin to a learnable and class-related parameter. The Equ. 3 can be modified as:

$$L_{ad} = -\frac{1}{M}\sum_{j=1}^{M}\log\frac{e^{s(\cos(\theta_{y^{(j)}j})-m_{y^{(j)}})}}{e^{s(\cos(\theta_{y^{(j)}j})-m_{y^{(j)}})}+\sum_{i=1,i\neq y^{(j)}}^{N} e^{s\cos(\theta_{ij})}} \tag{4}$$

where the $m_{y^{(j)}}$ is the margin corresponding to class $y^{(j)}$. Intuitively, we prefer larger m to reduce the intra-class variations. In this work, we constrain margins in the database view:

$$L_m = -\frac{1}{N}\sum_{i=1}^{N} m_i \tag{5}$$

which is the average of margins of all the classes. Combining the two parts is our Adaptive Margin Softmax Loss (AdaM-Softmax):

$$L_{AdaM} = L_{ad} + \lambda * L_m \tag{6}$$

The λ controls the strength of the margin constraint L_m, which is discussed in the experiments. Note that without L_m, there exists a trivial solution that $m_i = 0$. The proposed adaptive margin can be applied in any margin based Softmax loss such as ArcFace, just by changing the cosine margin to the angular margin.

3.1.3 Comparison with Other Loss Functions

To better understand the difference between our method and other margin based Softmax loss, we give the decision boundaries under the binary classification case in Table 1 and Figure 4. The main difference among these methods is our margin is learnable and class-related. As can be seen in Figure 4, although CosFace and ArcFace give a clear margin between the two classes, for the poor class C_1, its real distribution may be larger than the observed distribution, so that the real margin is getting smaller, leading to poor generalization. Differently, for AdaM-Softmax, it can learn a larger m_1 for C_1 through the parameter update during network training, making the observed features of C_1 more compact and implicitly pushing the real boundary of C_1 away from C_2.

In addition, to intuitively visualize the effect of AdaM-Softmax, we designed a toy experiment to demonstrate the feature distributions trained by different loss functions. We select face images from 8 identities in MS-Celeb-1M to train several 10-layer ResNet models which output 3-dimentional features. Among them, class 0 (red) contains the most (more than 400) samples, class 1, 2 (orange, gold) contain rich (about 200) samples, class 3∼7 (five cold colors) contain poor (about 50) samples (this ratio roughly simulates the sample number distribution of MS-Celeb-1M). We normalized the obtained 3-dimensional features and plot them on the sphere. The participated losses are Softmax loss, CosFace, and the proposed AdaM-Softmax with different λ. As shown in Figure 5, we can observe that Softmax loss prefers rich classes (such as class 0) and allocates large spaces for them, leading to bad decision boundaries. CosFace reduces intra-class variations and allocates equal

space for each class, without considering its sample distribution. For example, light blue points and red points occupy almost the same space area. The AdaM-Softmax focuses on optimizing the poor classes (cold colors) to be more compact. By comparing CosFace and AdaM-Softmax ($\lambda = 5$), we can see that the area occupied by the rich class 0 (red points) is almost the same, while for poor classes (blue, light blue and purple), the features of our method are more compact. Furthermore, by increasing the λ, the features of those samples of poor classes are almost clustered at one point.

Loss Functions	Decision Boundaries
Softmax	$(\mathbf{w}_1 - \mathbf{w}_2)\,\mathbf{x} + b_1 - b_2 = 0$
W-Norm Softmax	$\|\mathbf{x}\|\,(\cos\theta_1 - \cos\theta_2) = 0$
SphereFace [16]	$\|\mathbf{x}\|\,(\cos m\theta_1 - \cos\theta_2) = 0$
F-Norm SphereFace	$s(\cos m\theta_1 - \cos\theta_2) = 0$
CosFace [32, 34]	$s(\cos\theta_1 - m - \cos\theta_2) = 0$
ArcFace [4]	$s(\cos(\theta_1 + m) - \cos\theta_2) = 0$
AdaM-Softmax(CosFace)	$s(\cos\theta_1 - m_1 - \cos\theta_2) = 0$
AdaM-Softmax(ArcFace)	$s(\cos(\theta_1 + m_1) - \cos\theta_2) = 0$

Table 1. Decision boundaries for class 1 under binary classification case. Note that, θ_i is the angle between $\mathbf{w}_i$ and $\mathbf{x}$, s is the scale factor, m is the constant margin, and m_1 is the margin of class C_1.

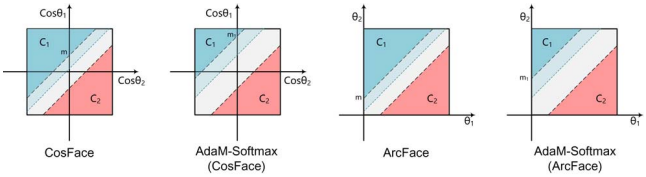

Figure 4. The comparison of decision margin of different loss functions for two classes. C_1 is a poor class and C_2 is a rich class. The black dashed line represents decision boundary of the observed samples in the database, the blue dashed line represents underlying real decision boundary of all the possible samples of C_1, and the gray areas are decision margins.

3.2. Hard Prototype Mining

Sampling has been extensively researched in verification losses (contrastive [27], triplet [24]), which is the hard example mining strategy [24, 21, 10]. The hard example mining aims to mine the most valuable pairs or triplets. In order to apply the idea of hard example mining in Softmax loss to improve efficiency and performance, we propose the Hard Prototype Mining (HPM) to select the classes that are most similar to the samples in the mini-batch in each iteration. The HPM is improved from the prototype selection strategy in [47]. Specifically, we consider the weight $\mathbf{w}_i$ of each class as its prototype. We build an ANN graph for prototypes of all classes and find the k classes that are most similar to each class and put them into their respective queues. We call these queues dominant queues indicated by $\mathcal{Q}_i$. When each iteration starts, we select the prototypes in the dominant queues corresponding to the samples in the mini-batch to construct the weight matrix W of this iteration. After forward propagation, we update the dominant queues by the scores calculated by inner products in the classification layer. Firstly, for a feature $\mathbf{x}_j$, if its highest activated class c_p is its corresponding class $y^{(j)}$, there is no need to update. Secondly, if $c_p \neq y^{(j)}$, we find all the classes with scores greater than $\cos(\theta_{y^{(j)}j})$ by sorting to update the queue. Finally, different from [47], we set a hyper-parameter h to control the size of dominant queue of each class. For each class in the queue, if its similarity to the queue owner is greater than h, it will remain in the queue, otherwise it will be popped up. With h, we can control the similarity of the selected prototypes, and gradually increase the difficulty of the training by adjusting h. The whole HPM increases little computation cost.

3.3. Adaptive Data Sampling

When the network has roughly converged, most of the samples in the dataset have been well classified and are difficult to contribute to the network training. To improve training effectiveness and efficiency, we build a feedback channel from the classification layer to the data layer to adaptively find valuable samples to form mini-batches, and we call it Adaptive Data Sampling (ADS). Specifically, we assign sampling probability to each sample. During training, when the sample is correctly classified in this iteration, we pass the signal to the data layer and reduce its sampling probability. Otherwise, we increase its sampling probability, so that the samples which are correctly classified frequently will be gradually ignored as the training progresses. We also set a minimum sampling probability s_{min}, in case those simple samples are never sampled.

Besides, since large-scale face data inevitably has much noise data [42], as training proceeds, the noisy samples will be continuously misclassified and have large sampling probability. In order to alleviate the impact of noisy data, we add feedback for noisy samples. For each sample in mini-batch, if the score between its feature and its corresponding prototype is lower than a threshold, we will pass the message to the data layer to drastically reduce the sampling probability of this sample.

4. Experiments

4.1. Experimental Settings

Preprocessing We detect faces by the FaceBox [43] detector and localize 5 landmarks (two eyes, nose tip and two mouth corners) by a simple 6-layer CNN. All the faces are normalized by similarity transformation and cropped to 120×120 RGB images.

CNN Architecture PyTorch [1] is used to implement

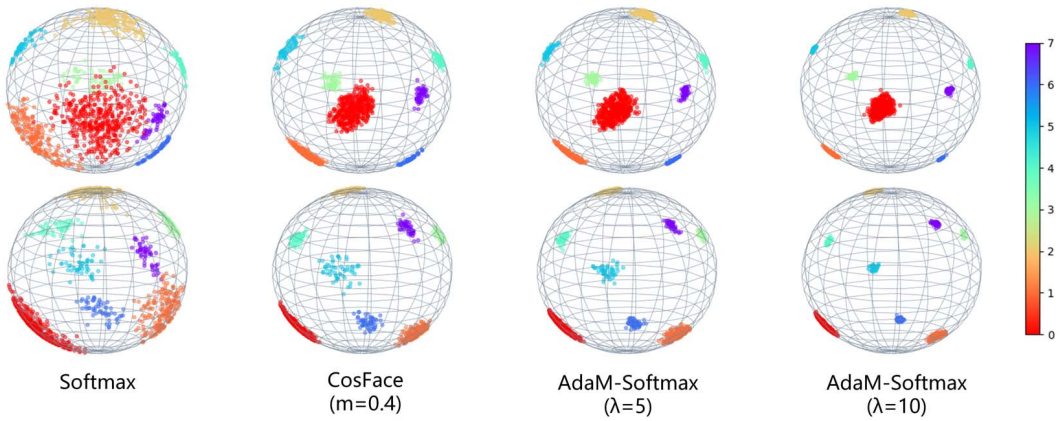

Figure 5. Feature distribution visualization of several loss functions. For better viewing, we show two perspectives of the same sphere. The first line focuses on the sample-rich classes (red, orange and gold), and the second line highlights those classes that are lacking in the sample (cold colors).

our proposed methods. All CNN models in the experiments follow the same architecture in this paper, which is a 50-layer residual network [8] same as LResNet50A-IR in [4]. It has four residual blocks and finally gets a 512-dimensional feature by average pooling. The networks are trained on TITANX GPUs and the batch size is set to fill all the GPU memory.

Training Data For all models in this paper, we trained them on the MS-Celeb-1M dataset [5], which is one of the largest wild dataset containing $98,685$ celebrities and 10 million images. Since there are much noise, the data is cleaned by the list of [38]. There are $79,077$ identities and 5 million images remaining. These face images are horizontally flipped for data augmentation.

Evaluation Setup For each image, we extract features only from the original image as the final representation. We didn't extract features from both the original image and the flipped one and concatenate them as the final representation. The score is measured by the cosine distance of two features. Finally, face verification and identification are conducted by thresholding and ranking the scores. We evaluate our models on LFW [12], LFW BLUFR [14] and MegaFace [13].

4.2. Overall Benchmark Comparisons

4.2.1 Experiments on MegaFace

MegaFace [13] is one of the most challenging testing benchmark for large-scale face identification and verification, which aims to evaluate the performance of face recognition models at the million scale of distractors. The gallery set of MegaFace is a subset of Flickr photos, composed of more than one million face images. The probe sets are two existing databases: FaceScrub [20] and FGNet. The FaceScrub dataset contains 106,863 face images of 530 celebrities. The FGNet dataset is mainly used for testing age invariant face recognition, with 1002 face images from 82 persons. In this study, we use the FaceScrub dataset as the probe set to evaluate the performance of our approach on MegaFace Challenge 1. However, there are some noisy images from FaceScrub and MegaFace, hence we use the noises list proposed by ArcFace [4] to clean it. For fair comparison, we implement the Softmax, A-Softmax, CosFace, ArcFace and our AdaptiveFace with the same 50-layer CNN. Table 2 shows the results of our models trained on the protocol of MegaFace large. The proposed AdaptiveFace obtains the best performance on both identification and verification tasks, compared with related baseline methods including Softmax, SphereFace, CosFace and ArcFace. Comparing with CosFace, AdaptiveFace improves the MF1 Rank1 from 93.942% to **95.023%** and boost the MF1 verification performance from 94.115% to **95.608%**, demonstrating the effectiveness of our method.

4.2.2 Experiments on LFW and LFW BLUFR

LFW [12] is a common face verification testing dataset in unconstrained conditions. It includes 13,233 face images from 5749 identities collected from the website with large variations in pose, expression and illumination. We follow the standard protocol of unrestricted with labeled outside data [11] to evaluate our model, and report the result on the 6,000 pair testing images from LFW. As shown in Table 3, AdaptiveFace improves the performance from 99.53% to **99.62%** on LFW. Considering that LFW has been well solved, we further evaluate our method on the more challenging LFW BLUFR protocol [14], which focuses on low FARs. We report the result in Table 4. As can be seen that our method is superior to all current state-of-the-art methods.

Method	Protocol	MF1 Rank1	MF1 Veri.
Beijing FaceAll_Norm_1600	Large	64.80	67.11
Google - FaceNet v8[24]	Large	70.49	86.47
NTechLAB - facenx_large	Large	73.30	85.08
SIATMMLAB TencentVision	Large	74.20	87.27
DeepSense V2	Large	81.29	95.99
YouTu Lab	Large	83.29	91.34
Vocord - deepVo V3	Large	91.76	94.96
CosFace[34]	Large	82.72	**96.65**
Softmax	Large	71.366	73.048
SphereFace[16]	Large	92.241	93.423
CosFace[34]	Large	93.942	94.115
ArcFace[4]	Large	94.637	94.850
AdaptiveFace	Large	**95.023**	**95.608**

Table 2. Face identification and verification evaluation on MF1. "Rank 1" refers to rank-1 face identification accuracy and "Veri." refers to face verification TAR under 10^{-6} FAR.

Method	Training Data	#Models	LFW
Deep Face[30]	4M	3	97.35
FaceNet[24]	200M	1	99.63
DeepFR [22]	2.6M	1	98.95
DeepID2+[29]	300K	25	99.47
Center Face[36]	0.7M	1	99.28
Baidu[15]	1.3M	1	99.13
SphereFace[16]	0.49M	1	99.42
CosFace[34]	5M*	1	**99.73**
Softmax	5M	1	98.83
SphereFace[16]	5M	1	99.57
CosFace[34]	5M	1	99.53
ArcFace[4]	5M	1	99.57
AdaptiveFace	5M	1	**99.62**

Table 3. Face verification (%) on the LFW datasets. "#Models" indicates the number of models that have been used in the method for evaluation. "*" indicates although the dataset of CosFace used also contains 5M images, it is composed of several public datasets and a private face dataset, containing about more than 90K identities, which is different with the dataset we used.

Method	VR@FAR =0.001%	VR@FAR =0.01%	DIR@FAR=1%
Softmax	83.41	93.55	80.16
SphereFace[16]	97.18	99.12	96.72
CosFace[34]	98.27	99.35	97.76
ArcFace[4]	98.48	99.47	98.02
AdaptiveFace	**98.89**	**99.53**	**98.19**

Table 4. The performance (%) on LFW BLUFR protocol.

4.3. Ablation Study

To demonstrate the effectiveness of three components in our framework, we run a number of ablations to analyze the improvements from AdaM-Softmax, Hard Prototype Mining and Adaptive Data Sampling, respectively. The baseline is CosFace when none of them is adopted. From Table 5, we can see that improvement from AdaM-Softmax is most obvious (from 94.115% to **95.032%** in MF1 Veri.). ADS and HPM can also improve performance when combined with AdaM-Softmax from 94.373% to **95.023%** in MF1 Rank1. When the three parts are combined, AdaptiveFace has a significant improvement over CosFace in all the evaluations.

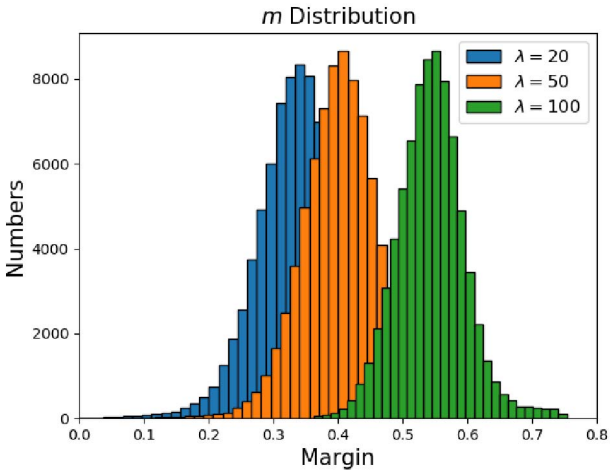

Figure 6. The margin distributions with different λ. The larger λ, the larger the mean of the distribution.

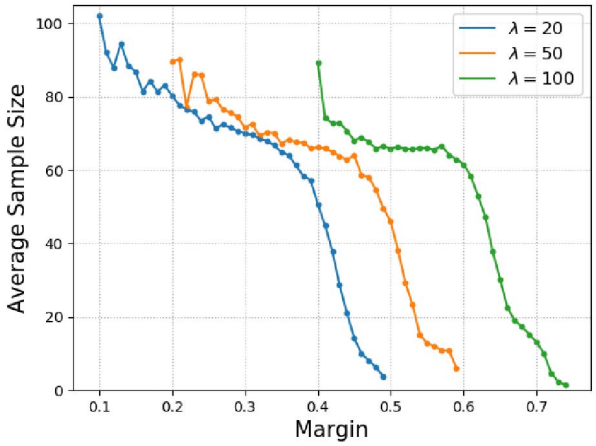

Figure 7. The average sample number of corresponding classes in different m and with different λ. The larger margin is, the smaller the number of samples of the corresponding class.

4.4. Exploratory Experiments

Effect of λ in Adaptive Margin Softmax Loss. Adaptive Margin Softmax Loss consists of two parts, classification loss L_{ad} and margin average loss L_m. The second part plays an important role to prevent m_i from getting smaller and smaller during training. In this part, we conduct an experiment to explore its impact. By varying λ from 0 to 150, we used MS-Celeb-1M to train our model and validate it on LFW, LFW BLUFR and MegaFace. The initial value of

ADS	HPM	AdM-Soft	BLUFR VR@FAR=0.01%	BLUFR VR@FAR=0.001%	MF1 Rank 1	MF1 Veri.
-	-	-	99.35	98.27	93.942	94.115
✓	-	-	99.39	98.44	94.068	94.539
-	✓	-	99.39	98.47	94.049	94.789
-	-	✓	99.47	98.53	94.373	95.032
✓	✓	✓	**99.53**	**98.89**	**95.023**	**95.608**

Table 5. Ablation study on LFW BLUFR and MegaFace. ADS indicates the Adaptive Data Sampling. HPM indicates the Hard Prototype Mining. AdM-Soft indicates the Adaptive Margin Softmax.

λ	LFW	BLUFR VR@FAR=0.001%	MF1 Rank 1	MF1 Veri.
CosFace[34]	99.53	98.27	93.942	94.115
0	99.45	98.29	93.138	93.348
20	99.48	98.40	94.074	93.921
50	99.53	98.53	94.373	**95.032**
70	99.53	**98.61**	94.590	94.687
100	**99.58**	98.47	**94.629**	94.641
150	99.48	98.55	94.613	94.250

Table 6. Performance (%) of Adaptive Margin Softmax with different λ on LFW, BLUFR and MegaFace.

m for all classes is 0.4. As shown in Table 6, we can see that the performances on LFW and MegaFace Rank1 are improved with the increasing of λ, and get saturated when $\lambda = 100$. While for verification in BLUFR and MegaFace, the performance is first increased, reaching the highest at $\lambda = 50$ or 70, then slightly decreasing. To further investigate the margins of each class under different λs, we plot the distribution of m at $\lambda = 20, 50$ and 100 in Figure 6. It can be seen that the distributions of m under different λs have an approximate Gaussian distribution with similar standard deviation, except that the mean of the distribution is increased by larger λ. In Figure 7, we show the average sample number of classes corresponding to the learned margin m. It can be found that as the number of samples decreases, the value of m is increasing, which validates that our AdaM-Softmax can adaptively allocate large margins to poor classes and allocate small margins to rich classes. It is obvious that the network can adaptively learn the margin of each class based on the sample distribution to handle the unbalanced data.

Effect of threshold h in Hard Prototype Mining. To explore the effect of similarity threshold h in our Hard Prototype Mining (HPM) approach, we train models with different h from small to large and compare their performance on LFW and LFW BLUFR protocol. The loss function we used in this experiment is the CosFace. Table 7 shows the results and the number of prototypes selected with different h, where $h = 0$ means we do not use HPM, i.e. training directly using CosFace. We can see that the threshold h can reduce the number of prototypes selected in each iteration and improve the final performance. Note that when $h = 0.23$, the selected prototypes are insufficient and too difficult, leading to degraded performance.

h	Number of Prototypes	LFW	BLUFR VR@FAR=0.01%	BLUFR VR@FAR=0.001%
0	79,077	99.53	99.35	98.27
0.1	20,000	99.57	99.38	98.33
0.15	10,000	**99.62**	**99.42**	98.37
0.2	5,000	99.52	99.39	**98.47**
0.23	2,000	99.50	99.19	97.72

Table 7. Performance (%) of HPM with different threshold h and the number of prototypes selected by different h.

5. Conclusion

In this paper, we proposed a novel approach AdaptiveFace for face recognition, which consists of three components. The first is the AdaM-Softmax, which introduces the adaptive margin for each class to adaptively minimize intra-class variances. The second is the Hard Prototype Mining, aiming to make the model concentrate on hard classes by adaptively selecting a small number of hard prototypes. The last one is Adaptive Data Sampling, which adaptively finds valuable samples through a feedback channel from the classification layer to the data layer. Our approach has seen significant improvements in several face benchmarks as depicted in experiment section. We believe that our approach could be very helpful for large-scale unbalanced data training in practice.

6. Acknowledgements

This work has been partially supported by the Chinese National Natural Science Foundation Projects #61876178, #61806196, #61872367, #61572501 and Science and Technology Development Fund of Macau (Grant No. 0025/2018/A1 152/2017/A).

References

[1] Pytorch. https://pytorch.org/.

[2] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman. Vggface2: A dataset for recognising faces across pose and age. pages 67–74, 2018.

[3] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face

verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 539–546. IEEE, 2005.

[4] Jiankang Deng, Jia Guo, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. *arXiv preprint arXiv:1801.07698*, 2018.

[5] Yandong Guo, Lei Zhang, Yuxiao Hu, Xiaodong He, and Jianfeng Gao. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In *European Conference on Computer Vision*, pages 87–102. Springer, 2016.

[6] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006.

[7] Ben Harwood, BG Kumar, Gustavo Carneiro, Ian Reid, Tom Drummond, et al. Smart mining for deep metric learning. pages 2821–2829, 2017.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *computer vision and pattern recognition*, pages 770–778, 2016.

[9] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, pages 84–92. Springer, 2015.

[10] Chen Huang, Chen Change Loy, and Xiaoou Tang. Local similarity-aware deep feature embedding. In *Advances in Neural Information Processing Systems*, pages 1262–1270, 2016.

[11] 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*, pages 14–003, 2014.

[12] Gary B Huang, Marwan Mattar, Tamara Berg, and Eric Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. In *Workshop on faces in'Real-Life'Images: detection, alignment, and recognition*, 2008.

[13] 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*, pages 4873–4882, 2016.

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

[15] Jingtuo Liu, Yafeng Deng, Tao Bai, Zhengping Wei, and Chang Huang. Targeting ultimate accuracy: Face recognition via deep embedding. *arXiv preprint arXiv:1506.07310*, 2015.

[16] 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*, 2017.

[17] Weiyang Liu, Yandong Wen, Zhiding Yu, and Meng Yang. Large-margin softmax loss for convolutional neural networks. In *ICML*, pages 507–516, 2016.

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

[19] Aaron Nech and Ira Kemelmacher-Shlizerman. Level playing field for million scale face recognition. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3406–3415. IEEE, 2017.

[20] Hong-Wei Ng and Stefan Winkler. A data-driven approach to cleaning large face datasets. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 343–347. IEEE, 2014.

[21] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4004–4012, 2016.

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

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

[24] 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.

[25] Evgeny Smirnov, Aleksandr Melnikov, Sergey Novoselov, Eugene Luckyanets, and Galina Lavrentyeva. Doppelganger mining for face representation learning. In *International Conference on Computer Vision*, 2017.

[26] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems*, pages 1857–1865, 2016.

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

[28] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep learning face representation from predicting 10,000 classes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1891–1898, 2013.

[29] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deeply learned face representations are sparse, selective, and robust. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2892–2900, 2015.

[30] 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*, pages 1701–1708, 2014.

[31] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Web-scale training for face identification. pages 2746–2754, 2015.

[32] Feng Wang, Weiyang Liu, Haijun Liu, and Jian Cheng. Additive margin softmax for face verification. *Signal Processing Letters, IEEE*, 25:926–930, 2018.

[33] Feng Wang, Xiang Xiang, Jian Cheng, and Alan Loddon Yuille. Normface: l 2 hypersphere embedding for face verification. pages 1041–1049, 2017.

[34] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. In *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*. IEEE, 2018.

[35] Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. Learning fine-grained image similarity with deep ranking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1386–1393, 2014.

[36] 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.

[37] Chao-Yuan Wu, R Manmatha, Alexander J Smola, and Philipp Krähenbühl. Sampling matters in deep embedding learning. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2017.

[38] Xiang Wu, Ran He, Zhenan Sun, and Tieniu Tan. A light cnn for deep face representation with noisy labels. *IEEE Transactions on Information Forensics and Security*, 13(11):2884–2896, 2018.

[39] Qiqi Xiao, Hao Luo, and Chi Zhang. Margin sample mining loss: A deep learning based method for person re-identification. *arXiv preprint arXiv:1710.00478*, 2017.

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

[41] Yuhui Yuan, Kuiyuan Yang, and Chao Zhang. Hard-aware deeply cascaded embedding. pages 814–823, 2017.

[42] Xiaohang Zhan, Ziwei Liu, Junjie Yan, Dahua Lin, and Chen Change Loy. Consensus-driven propagation in massive unlabeled data for face recognition. pages 568–583, 2018.

[43] Shifeng Zhang, Xiangyu Zhu, Zhen Lei, Hailin Shi, Xiaobo Wang, and Stan Z Li. Faceboxes: A cpu real-time face detector with high accuracy. pages 1–9, 2017.

[44] Xiao Zhang, Zhiyuan Fang, Yandong Wen, Zhifeng Li, and Yu Qiao. Range loss for deep face recognition with long-tailed training data. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[45] Yutong Zheng, Dipan K Pal, and Marios Savvides. Ring loss: Convex feature normalization for face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5089–5097, 2018.

[46] Jia-Xing Zhong, Ge Li, and Nannan Li. Deep metric learning with false positive probability. In *International Conference on Neural Information Processing*, pages 653–664. Springer, 2017.

[47] Xiangyu Zhu, Hao Liu, Zhen Lei, Hailin Shi, Fan Yang, Dong Yi, Guojun Qi, and Stan Z. Li. Large-scale bisample learning on id versus spot face recognition. *International Journal of Computer Vision*, 2019.