Fair Loss: Margin-Aware Reinforcement Learning for Deep Face Recognition

Bingyu Liu, Weihong Deng*, Yaoyao Zhong, Mei Wang, Jiani Hu
Beijing University of Posts and Telecommunications
{liubingyu,whdeng,zhongyaoyao,wangmei1,jnhu}@bupt.edu.cn

Xunqiang Tao, Yaohai Huang
Canon Information Technology (Beijing) Co., Ltd
{taoxunqiang,huangyaohai}@canon-ib.com.cn

Abstract

Recently, large-margin softmax loss methods, such as angular softmax loss (SphereFace), large margin cosine loss (CosFace), and additive angular margin loss (ArcFace), have demonstrated impressive performance on deep face recognition. These methods incorporate a fixed additive margin to all the classes, ignoring the class imbalance problem. However, imbalanced problem widely exists in various real-world face datasets, in which samples from some classes are in a higher number than others. We argue that the number of a class would influence its demand for the additive margin. In this paper, we introduce a new margin-aware reinforcement learning based loss function, namely fair loss, in which each class will learn an appropriate adaptive margin by Deep Q-learning. Specifically, we train an agent to learn a margin adaptive strategy for each class, and make the additive margins for different classes more reasonable. Our method has better performance than present large-margin loss functions on three benchmarks, Labeled Face in the Wild (LFW), Youtube Faces (YTF) and MegaFace, which demonstrates that our method could learn better face representation on imbalanced face datasets.

1. Introduction

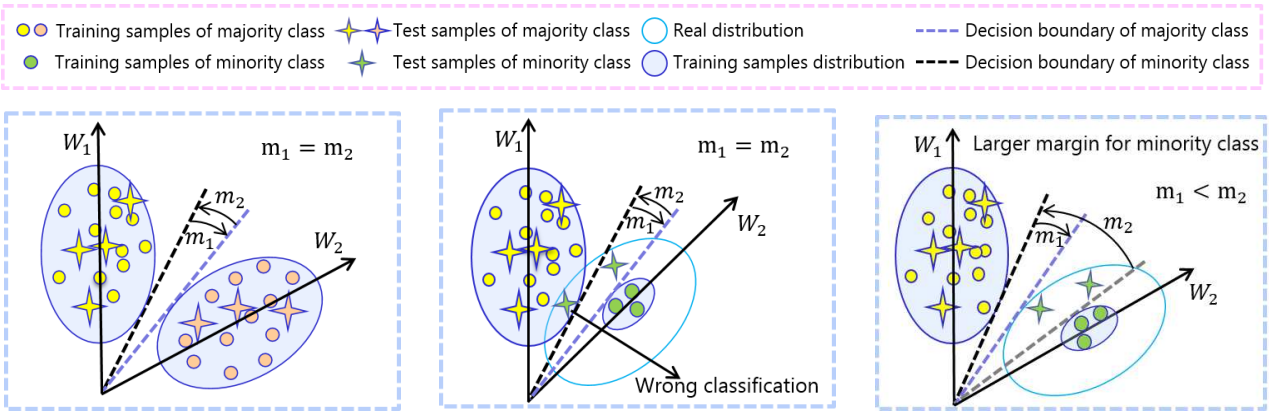
Deep convolutional neural networks (DCNNs) [28, 17, 10, 31, 43, 11, 37, 18] have become the mainstream technique for deep face recognition, which made remarkable progress on both recognition accuracy and robustness. Deep face models are usually learned in a supervised manner by various loss functions. Classical softmax loss function is the most widely used for generic objects, but it is not discriminative enough for face recognition due to the fine-grained property [40, 21, 20]. To address this limitation, several large-margin loss functions have been proposed to improve the generalization ability of softmax loss, such as SphereFace [20], CosFace [36], and ArcFace [5]. These losses successfully improve the generalization ability of DCNN by incorporating a fixed margin to softmax, and achieve state-of-the-art performance on major face benchmarks, such as LFW [13], YTF [41] and MegaFace [14, 24].

However, as illustrated in Fig. 1, previous large margin methods ignore the class imbalance of the training data, which refers to that the samples of majority classes are much more than those of minority classes. Deep face models are inevitably affected by the class imbalance problem. Owing to the lack of intra-class variance, the minority classes often fail to describe the real feature space. In this situation, the fixed additive margin for both majority and minority classes will aggravate the biased decision boundary. Fig. 1(b) shows that the model tends to make mistakes on new test samples of minority classes, which reflects the bad generalization ability of deep representation. As a result, a fixed margin may be misleading for unbalanced classes.

To improve the large-margin loss, we dedicate to explore adaptive boundaries between various classes, considering their representativeness to characterize the actual distribution. Majority classes would not converge with an excessively large margin, while a minority class would be squeezed toward majority classes given a relatively small margin. We propose to balance the additive margins between various classes. As shown in Fig. 1(c), a majority class requires a relatively smaller margin, while a minority class needs a relatively larger margin. By the adaptive margin, the test accuracy is significantly boosted. Meanwhile, we assume that the demand for the margin of a particular class may be different in the training progress following the change of training state, such as the intra-class variance.

Inspired by this observation, we propose a new fair loss with a margin adaptive strategy by reinforcement learning. Specifically, we simulate all kinds of changes of additive

*Corresponding Author



(a) Fixed margins between majority classes

(b) Fixed margins between majority class and minority class

(c) Adaptive margins between majority class and minority class

Figure 1. Effects of class imbalance. (a) A fixed additive margin is quite appropriate for two majority classes. (b) The model tends to make mistakes on new test samples of a minority class with the fixed additive margin. (c) We propose that the demands for additive margins should be different: a majority class needs a relatively smaller margin while a minority class needs a relatively larger margin.

margins for classes in the training process and collect the influence on the training model, from which we conclude a universal strategy of the adaptive margin by using Deep Q-learning [23]. Eventually this strategy is generalized in other models training progress, guiding a class to adapt its additive margin according to a specific training state.

Our contributions can be summarized as follows:

(1) We propose a new fair loss function that takes the prevalent class imbalance problem into consideration to learn adaptive margins. By fully considering the distributional property of neighbor classes, fair loss displays more comprehensive margin setting for the recognition problem.

(2) We successfully apply reinforcement learning to optimize the fair loss by training an agent to learn a margin adaptive strategy. We empirically demonstrate that the strategy can be applied to various models with large margin, which confirms the generalization ability of our method.

(3) Extensive experiments on LFW [13], YTF [41], and MegaFace [14, 24] show that our method achieves state-of-the-art performance on face recognition. Compared with other large-margin methods, it has expressive superiority on learning discriminative features from imbalanced datasets.

2. Related Work

Large-margin loss functions. Deep face recognition has become a hot spot thanks to the advancement of DCNNs. Loss function plays a major role in CNN models. The softmax loss function is commonly used as the supervision signal in face recognition. However, softmax loss is not effective to learn discriminative representation as training on million level identities so that intra-class variations could be larger than inter-class distances. Recently, the mainstream method is to use a large-margin loss function [20, 36, 5] (based on traditional softmax loss function) to train a feature extractor to make features more discriminative. Liu *et al*. [20] propose A-softmax (SphereFace) by introducing a multiplicative angular margin to softmax loss and make the decision regions become more separated. Wang *et al*. [36] propose large margin cosine loss (CosFace) to further maximize the decision margin in the cosine space. CosFace overcomes the optimization difficulty of SphereFace and is much easier to reproduce. Deng *et al*. [5] directly add an angular margin in the angular space and have a more clear geometric interpretation. These large-margin loss functions all improve the softmax loss by incorporating a fixed margin. However, they ignore the class imbalance problem.

Deep imbalanced learning. To tackle the class imbalance problem, the original efforts can be mainly classified into two groups: data re-sampling [9, 2, 6, 7, 8] and cost-sensitive learning [16, 33, 35, 50]. In the last few years, several works [49, 38, 15, 39] have addressed imbalanced learning via deep models. However, as for the task of face recognition, few methods focus on class imbalance problem. Yin *et al*. [45] adapt the distribution of learned features from minority classes to mimic that of majority classes by augmenting the minority classes in feature space. Wu *et al*. [42] propose a center invariant loss which aligns the feature centers of the minority classes to the majority. Zhang *et al*. [48] propose a range loss minimizing the ranges (the largest intra-class distances) to enhance the model's learning ability towards minority classes. Unfortunately, all these methods cannot guarantee the high discriminative power between all classes. Our method complements them by providing a fair loss with adaptive margins that enable imbalanced classes to have appropriate margins, which cannot only avoid minority classes being submerged but also increase the discrimination of features.

Reinforcement learning (RL). RL trains an agent to learn policies based on trial and error in a dynamic environment by maximizing the accumulated reward. Deep RL

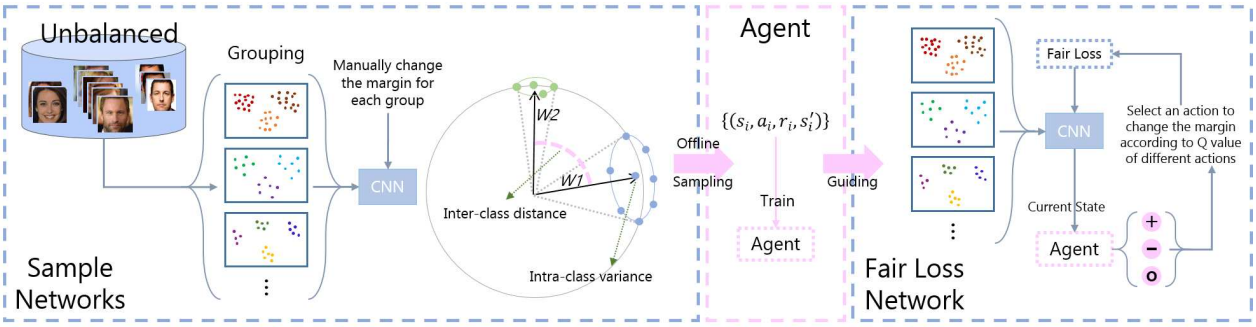


Figure 2. An illustration of our proposed method. Firstly, we train a series of sample networks by manually changing the margin in loss to collect samples for the agent, represented by $\{(s_i, a_i, r_i, s_i')\}$. Details of $\{(s_i, a_i, r_i, s_i')\}$ refer to section 3.2 and Fig. 3. Then we use the samples to train the agent for a margin adaptive strategy, which will give action outputs for state inputs. Finally, we train our fair loss network with changes of margins by the action outputs from the agent.

with CNNs has achieved human-level performance in Atari Games. In addition to its traditional applications in robotics and control, recently RL has been successfully applied to a few visual recognition tasks. RL is employed to object tracking, researched in [12]. An agent is trained to learn when to stop advancing the features to the next layer and make prediction, which achieves a remarkable balance between tracking speed and accuracy. To deal with multi-shot pedestrian re-identification problem, Zhang *et al*. [46] train an agent to stop the comparison after receiving sufficient pairs, which obtain competitive performance with the state-of-the-art methods while using much fewer images. In our work, RL is used to learn a margin adaptive strategy for diverse training identities.

3. Proposed Approach

The overview of our method is depicted in Fig. 2. First, we train a CNN by manually changing the margin in the loss to collect a series of samples we define. Then we use the samples to train an agent for margin adaptive strategy, which will give action output for state input. Finally, we train our fair loss network with margins changing by the action outputs from the agent. In this section, we will introduce the details of our approach. We first start with the proposed fair loss function. Then we introduce the adaptive margin in the fair loss via reinforcement learning, including the formulation of our sample space, the training of the agent and the application of the margin adaptive strategy.

3.1. Fair Loss

We define the proposed fair loss function as follows:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{P_{y_i}^*(m_i(t), x_i)}{P_{y_i}^*(m_i(t), x_i) + \sum_{j=1, j\neq y_i}^{n} P_j(x_i)}, \tag{1}$$

where $x_i \in \mathbb{R}^d$ denotes the deep feature of the i-th sample, belonging to the y_i-th class. The batch size and the class number is N and n, respectively.

In the function P^*, we have an adaptive margin $m_i(t)$. Compared to other large-margin loss functions, we make m vary during the training instead of remaining unchanged. In fact, different classes have different demands for the margins, and the demands may change during the training. Our adaptive margin $m_i(t)$ depends on i and is a function of t, where t represents the stage of the training.

The formulation of P can be represented as follows:

$$P_j(x_i) = e^{s\cos\theta_j}, \tag{2}$$

subject to

$$W_j = \frac{W_j}{\|W_j\|}, x_i = \frac{x_i}{\|x_i\|}, \cos\theta_j = W_j^T x_i. \tag{3}$$

$W_j \in \mathbb{R}^d$ denotes the j-th column of the weight $W \in \mathbb{R}^{d\times n}$ in the last fully connected layer.

Considering the formulation of the function P^*, ours is based on other large-margin loss functions, *e.g*. CosFace (LMCL) [36] and ArcFace [5]. Specifically, based on CosFace, the function P^* can be formulated as follows:

$$P_{y_i}^*(m_i(t), x_i) = e^{s(\cos(\theta_{y_i}) - m_i(t))}, \tag{4}$$

and based on ArcFace, P^* can be formulated as follows:

$$P_{y_i}^*(m_i(t), x_i) = e^{s(\cos(\theta_{y_i} + m_i(t)))}, \tag{5}$$

where $\cos(\theta_{y_i})$ can be computed similarly to $\cos\theta_j$.

We fix $\|x_i\|$ by L2 normalization and re-scale $\|x_i\|$ to s, following [20, 36, 5]. In this paper, we use $s = 64$ for face recognition experiments.

3.2. Margin Adaptive Strategy Learning

We formulate the problem of finding an appropriate margin adaptive strategy as a Markov Decision Process (MDP), described by $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$ as the states, actions, transitions and rewards. We will train an agent to adjust the margin in every state. The agent will be fed with a series of samples,

which can be represented as $\{(s_i, a_i, r_i, s'_i)\}$. Here, $s_i \in \mathcal{S}$, $a_i \in \mathcal{A}$, $r_i \in \mathcal{R}$, and $s'_i \in \mathcal{S}$ means the next state which the agent turns to through the action a_i. After that, the trained agent will give us the margin adaptive strategy during the training of our fair loss network.

States: We consider that each class's demand for the margin can be affected by the number of images and the intra-class variance. Therefore, the state s_i includes three parts. The first part is equivalent of the adaptive margin m. In the second part, we divide all the classes into several groups according to the number of images and the intra-class variance. Here, the intra-class variance is obtained by using a trained neural network, which has a fixed margin loss function. We make $g(n_{y_i}, V^*_{y_i}) \in G$ represent the group, where $G = \{g_1, \cdots, g_{k_G}\}$. n_{y_i} denotes the number of images belonging to the y_i-th class. $V^*_{y_i}$ represents the intra-class variance, which can be formulated as follows:

$$V^*_{y_i} = \frac{1}{n_{y_i}} \sum_{j=1}^{n_{y_i}} \left\| x^*_j - \bar{x}^* \right\|^2, \tag{6}$$

where

$$\bar{x}^* = \frac{1}{n_{y_i}} \sum_{j=1}^{n_{y_i}} x^*_j. \tag{7}$$

x^*_j is the feature extracted by a pre-trained neural network and L2 normalized.

We also augment the current intra-class variance V_i as the third part. For each class in the group $g(n_{y_i}, V^*_{y_i})$, we calculate the intra-class variance by using the current neural network with our fair loss function and get the mean as V_i. The formulation can be represented as follows:

$$V_i = \frac{1}{N_i} \sum_{k=1}^{N_i} V'_k, \tag{8}$$

where N_i denotes the number of classes in the group $g(n_{y_i}, V^*_{y_i})$, and

$$V'_k = \frac{1}{n_{y_k}} \sum_{j=1}^{n_{y_k}} \|x_j - \bar{x}\|^2, \tag{9}$$

$$\bar{x} = \frac{1}{n_{y_k}} \sum_{j=1}^{n_{y_k}} x_j. \tag{10}$$

Here, x_j is the feature extracted by our current neural network and L2 normalized. In order to make the space of states discrete, we let $m_i(t) \in M$ and define a function $f(V_i) \in F$ to quantify V_i, where $M = \{m_1, \cdots, m_{k_M}\}$, $F = \{f_1, \cdots, f_{k_F}\}$. Therefore, s_i can be represented as $\{m_i(t), g(n_{y_i}, V^*_{y_i}), f(V_i)\}$.

Actions and Transitions: In every state, we have three actions for the agent: make m stay unchanged, make m add a fixed step and make m minus a fixed step. Our samples will include all of the actions and rewards associated with each action to train the agent for better decisions.

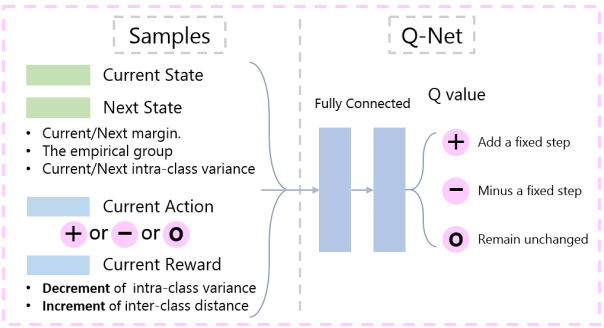


Figure 3. The training process of the agent. We use the samples collected by the sample networks to train the agent. Each sample includes four parts: current state, next state, action and reward, which can be represented by (s_i, a_i, r_i, s'_i). The Q-Net outputs Q-value of the three actions with a state input and the agent chooses the action with the largest Q-value as the output.

Rewards: We define the rewards for smaller intra-class variance and larger inter-class distance. Before that, we first define a function R_i, which is related to the state s_i and can be formulated as follows:

$$R_i = D_i - V_i. \tag{11}$$

D_i can be used to evaluate the inter-class distance, which can be formulated as follows:

$$D_i = \left\| W_{y_i} - W^*_{y_i} \right\|^2, \tag{12}$$

where $W^*_{y_i}$ denotes the weight vector of the class, which has the shortest distance from the y_i-th class. V_i is the same as defined by Eq. 8. Furthermore, reward r_i can be formulated as follows:

$$r_i = R'_i - R_i, \tag{13}$$

where R'_i is similar to R_i and related to the state s'_i.

Deep Q-learning: We use Deep Q-Learning [23] to make the agent find an optimal policy, since we do not have a priori knowledge about the correct action to choose. For each state and action (s_i, a_i), $Q(s_i, a_i)$ represents the discounted accumulated rewards for the state and action. During the training, we iteratively update the Q function by:

$$Q(s_i, a_i) = r_i + \gamma \max_{a'_i} Q(s'_i, a'_i). \tag{14}$$

The structure of the Q-network is illustrated in Fig. 3. We simply use a two-layer fully connected network as the Q function, with a hidden layer of 10 units. Each fully connected layer is followed by a ReLU activation function. Details of training the agent are summarized in Algorithm 2. Finally, the agent will output an action a_t using a policy $a_t = \arg\max_a Q(s_t, a)$, where s_t is the state representation above.

Algorithm 1 Collect samples for the agent

Preparation:

Pre-train a neural network with a stable margin loss function. Divide all the classes into several groups according to the number of images and the intra-class variance, which is calculated by the pre-trained network.

Collect Samples:

for g **in** all groups **do**

1: Calculate the current state s_i from the last network (initialized with the pre-trained network).

2: Do every action in the action space to modify the margin for group g.

3: For each action a_i, train the last network for another epoch by the adapted margin.

4: Calculate the state s_i', which group g turned to from the one-epoch training.

5: Calculate the reward r_i through the last network and the current one by Eq. 13.

6: Record the (s_i, a_i, r_i, s_i') as a sample.

if s_i' is a new state which hasn't appeared before **then**

7: **return** to step 1.

end if

end for

Algorithm 2 Train the agent for margin adaptive strategy

Data:

Collect samples $\{(s_i, a_i, r_i, s_i')\}$ from the original neural network by Algorithm 1.

Input:

A series of samples $\{(s_i, a_i, r_i, s_i')\}$.

Output:

The Q-Net parameters Ω.

Train:

while not reach the maximum iteration **do**

1: Forward pass the data (s_i, a_i).

2: Get the Q-Net output $Q(s_i, a_i)$.

3: For **each** action a_i' in the action space, forward pass (s_i', a_i').

4: Calculate the target output

$$y_i = r_i + \gamma \max_{a_i'} Q(s_i', a_i').$$

5: $\Omega \leftarrow \Omega - \alpha \sum_i \frac{\mathrm{d}Q(s_i, a_i)}{\mathrm{d}\Omega}(Q(s_i, a_i) - y_i)$.

end while

Samples Collecting: In order to train the agent, we need to collect samples to feed the agent. We first divide all the classes into several groups according to the number of images and the intra-class variance calculated by a pre-trained network with a stable large-margin loss function. Then we manually change the margin to train a series of sample networks and extract samples represented by $\{(s_i, a_i, r_i, s_i')\}$. The details are presented in Algorithm 1.

Algorithm 3 Train fair loss network

Preparation:

The trained agent.

Train:

1: Do the training as normal deep CNNs.

At the **end** of **every epoch**:

for g **in** all groups **do**

2: Calculate the current state s_g of group g.

3: Input the state s_g into the trained agent and get the output action a_g.

4: Do the action a_g to modify the margin for group g.

end for

5: Enter the next epoch of training.

Application of Margin Adaptive Strategy: The margin adaptive strategy generated by the trained agent will be used to conduct the training of our fair loss network by Algorithm 3. During the training, the agent will make decisions to modify the margins in fair loss for every group.

4. Experiments

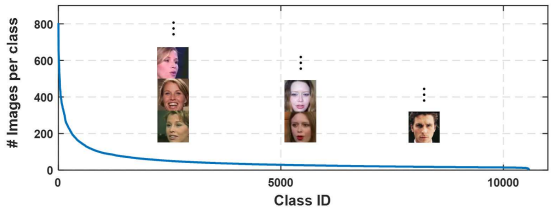
In this section, we will first introduce the implementation details in following experiments, and then describe the process of training the agent. After that, we design several validity experiments for our approach. Finally, we will present the results of our method on three open benchmarks, and compare them with other state-of-the-art methods.

4.1. Implementation Details

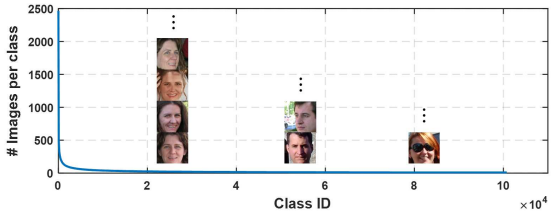
Preprocessing. We only use standard preprocessing. Face landmarks in all images for training and testing are detected by MTCNN [47]. Then, we adopt the 5 facial points to perform a similarity transformation. After that, the faces are cropped to 112×112. Following [20, 36], each pixel (in $[0, 255]$) in RGB images is normalized by subtracting 127.5 then being divided by 128.

Training. We use two datasets to train our models. One is a publicly available web-collected training dataset CASIA-WebFace [44], which contains 494,414 face images belonging to 10,575 different individuals. The scale of CASIA-WebFace is small (less than 0.5M images and 20K subjects) [14] so that we can use it to compare with the existing results of small training datasets. The other dataset is in large-scale, which is MF2 training dataset [24] provided by MegaFace. MF2 training dataset is used for MegaFace Challenge 2, containing 4.7 million faces and 672K unique identities. As shown in Fig. 4, both of the two datasets have imbalanced distribution.

As for the CNN architecture, we use two of the advanced networks. One is the same 64-layer CNN architecture as



(a) CASIA-WebFace dataset



(b) MF2 training dataset

Figure 4. Imbalanced data distribution of two training datasets.

described in [20] for Q-Net training samples collecting. The other is ResNet50 [10] with a modified structure, proposed in [5], after the last convolutional layer. It is used to conduct the experiments on several benchmarks. We use MxNet [4] to implement fair loss and the CNNs and TensorFlow [1] for Deep Q-learning. For the loss function, we make our fair loss use margin adaptive strategy based on CosFace (LMCL) [36] as an example. In fact, the proposed approach can also be used in other softmax-based loss functions which have margin parameters. CNN models are trained on two GPUs (NVIDIA GeForce 1080TI), setting the batch size as 256 for the small dataset and 200 for the large. On the small dataset, the learning rate begins with 0.1 and is divided by 10 at 100K, 140K iterations. The training process is finished at 180K iterations. On the large dataset, we divide the learning rate at 256K, 358K, 410K iterations and finish at 440K iterations. We set momentum at 0.9 and weight decay at 0.0005.

Testing. During testing, the score is computed by the cosine distance of two feature vectors, which are obtained by concatenating the original face features and the horizontally flipped features. At last, we use the nearest neighbor classifier and threshold comparison for face identification and verification, respectively. Our models are assessed on several popular public face datasets, including LFW [13], YTF [41], and MegaFace [14, 24].

4.2. Agent Training Process

In order to use fair loss to train our models, we train an agent to generate margin adaptive strategy. We first pre-train a 64-layer CNN on CASIA-WebFace [44], using CosFace (LMCL) [36] with the margin set to 0.35. Depending on the number of images and the intra-class variance calculated by the pre-trained network, we divide the identities in CASIA-WebFace into 9 groups represented by a number from 0.1 to 0.9. We choose 50 and 150 as the thresholds of the number of images. As for the intra-class variance, we also determine two thresholds so that the number of identities in the three segments is similar to each other.

Then we collect samples for the agent by training a series of networks. In detail, for each group, we manually modify the margin by all the actions we define and train an epoch from the pre-trained network. The one-epoch training leads the group to another state, in which the group has a new margin. We keep modifying the margin from the current state by all the actions and training another epoch from the current network until the margin has traversed the entire margin space. In this study, we set the margins varying from 0.15 to 0.45 (0.15, 0.25, 0.35 and 0.45). We can obtain the intra-class variance and inter-class distance of each group from the trained networks. Further, we calculate the rewards of the network transfers and record the vectors, which represent the current state, next state, action and reward of each network, as our agent training samples. After that, we feed the samples to train the agent by Deep Q-learning [23].

The trained agent can output the Q-value of the three actions with a state input. We will choose the action with the largest Q-value to conduct the adjustment of the margin in fair loss during the training. Fig. 5 shows part of the margin adaptive strategy from the trained agent, from which we have three findings. First, the group with smaller classes has a greater tendency to raise the margin. Second, larger intra-class variance is more likely to lead to increasing the margin, and vice versa. Large intra-class variance usually reflects that a class is not trained sufficiently, so this phenomenon can be interpreted that larger margins are needed to optimize the class trained insufficiently. Third, for a specific group with specific intra-class variance, there may exist an optimal range of margins. The first finding is consistent with our theoretical analysis, while the second and third ones further promote understanding and increase the reasonability of our margin adaptive strategy.

4.3. Generalization and Validation Experiments

To show the generalization ability of our approach, we perform numerous validity experiments. Our agent is trained based on CosFace [36]. Nevertheless, we use the strategy generated by the agent to modify the margins on not only CosFace [36] but also ArcFace [5]. Meanwhile, we use different network architectures from the sample networks, including 34-layer and 18-layer CNN based on the residual unit proposed in [5]. As shown in Table 1, our margin adaptive strategy can be applied to various models with large margin, which confirms the generalization ability of our method.

We also design three baselines with rule-based margin strategies. The MIP baseline simply uses margins inversely proportional to the number of images of the classes. The VDM baseline makes the margins decided by their accord-

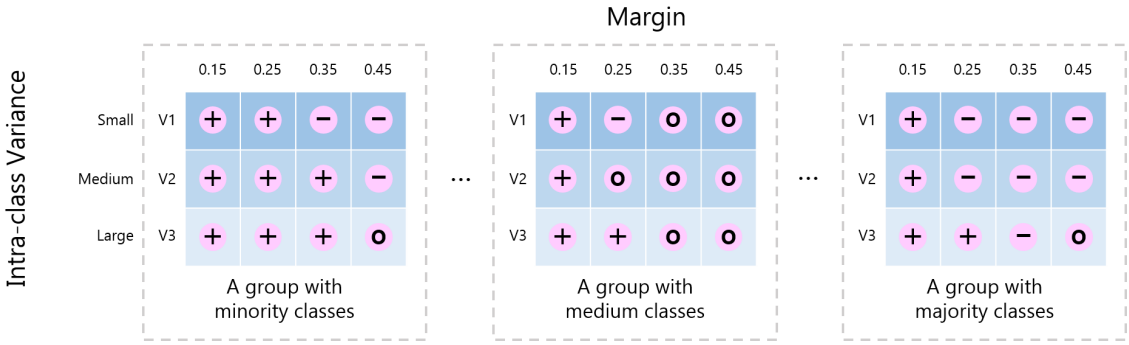


Figure 5. Part of the margin adaptive strategy from the trained agent. Each grid indicates an action on a state. "V1", "V2" and "V3" refer to small, medium and large intra-class variance, respectively. We have three findings from the strategy. First, the group with smaller classes has a greater tendency to raise the margin. Second, larger intra-class variance is more likely to lead to increasing the margin, and vice versa. Third, for a specific group with specific intra-class variance, there may exist an optimal range of margins.

Method	#layers	LFW	YTF	MF1 Rank1	MF1 Veri.
CosFace (LMCL) [36]	34	99.32	93.98	75.38	89.15
FairLoss_Cos	34	**99.52**	**95.82**	**76.95**	**90.82**
ArcFace [5]	34	99.42	94.54	75.93	91.06
FairLoss_Arc	34	**99.57**	**94.66**	**78.15**	**92.89**
CosFace (LMCL) [36]	18	99.23	93.58	73.56	86.80
FairLoss_Cos	18	**99.35**	**94.58**	**75.40**	**89.14**
ArcFace [5]	18	99.28	93.36	75.72	88.98
FairLoss_Arc	18	**99.37**	**93.76**	**75.98**	**90.32**

Table 1. Validation of our strategy with different loss functions and different network architectures on LFW [13], YTF [41] and the MegaFace Challenge 1 [14]. "Rank1" refers to rank-1 identification accuracy with 1M distractors, and "Veri." refers to verification TAR for 10^{-6} FAR. FairLoss_Cos and FairLoss_Arc represent the methods with our margin adaptive strategy used in CosFace [36] and ArcFace [5], respectively.

Method	LFW	YTF	MF1 Rank1	MF1 Veri.
CosFace (LMCL) [36]	99.33	96.1	77.11	89.88
CosFace_MIP	99.43	95.4	76.77	91.21
CosFace_VDM	99.37	95.8	76.78	91.43
CosFace_RM	99.33	95.6	76.09	91.36
FairLoss_Cos	**99.57**	**96.2**	**77.45**	**92.87**

Table 2. Comparisons with several baselines with rule-based margins. MIP simply uses margins inversely proportional to the number of images of the classes. VDM refers to variance deciding margins. RM uses random margins.

ing intra-class variances and inter-class variances during training. Smaller intra-class variance and larger inter-class variance lead to larger margin, and vice versa. The decision function is similar to our reward function in Eq. 13. The RM baseline uses random margins. For fair comparisons, we make the margins vary from 0.15 to 0.45 (0.15, 0.25, 0.35 and 0.45) both in our margin adaptive strategy and all the baselines. The results are given in Table 2. Our method outperforms all the three unfixed margin strategies, which verifies the effectiveness of our strategy.

In Fig. 6, we illustrate the distributions of average cosine distances between the weight of a class and that of other classes from the last fully connected layer. The classes are divided into three parts by the number of images with 50 and 150 as the thresholds. As can be observed, the distances of minority classes with our method are larger than the original CosFace [36] while smaller in majority classes. Note that the weight distance can reflect the margin for a same class in the two methods, so the distributions in Fig. 6 completely confirm our assumptions in Fig. 1(c). Further, the weights in our method are more orthogonal.

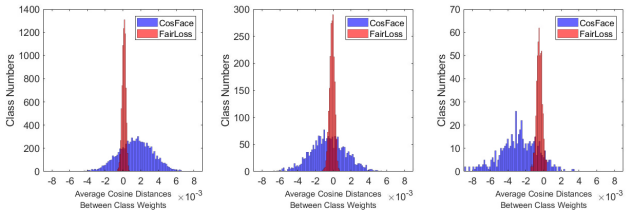


Figure 6. Distributions of average cosine distances between class weights. The classes in the three graphs have different number of images. Left: less than 50. Middle: between 50 and 150. Right: more than 150.

4.4. Experiments on LFW and YTF

Labeled Face in the Wild (LFW) dataset [13] contains 13,233 web-collected face images from 5,749 different identities. Youtube Faces (YTF) dataset [41] contains 3,425 videos of 1,595 different people, with 181.3 frames for the average length of a video clip. The faces in both datasets have huge variations in pose, expression and illuminations. Following the unrestricted with labeled outside data protocol on both datasets, we test on 6,000 face pairs from LFW and 5,000 video pairs from YTF.

As illustrated in Table 3, the proposed fair loss achieves 99.57% on LFW and 96.2% on YTF with single network trained on small dataset CAISA-WebFace [44]. It shows the best performance trained on WebFace, better than other models trained on the same dataset, including the original CosFace [36]. Further, compared with the recent imbalanced learning methods trained on WebFace or larger data (second cell in Table 3), fair loss is shown to achieve consistent gains due to the stronger discriminating power between imbalanced classes.

Method	#Nets	#Layers	Data	LFW	YTF
DeepFace [32]	3	6	4M	97.35	91.4
FaceNet [27]	1	14	200M	**99.63**	95.1
VGG Face [26]	1	16	2.6M	98.95	**97.3**
DeepID2+ [30]	25	-	0.3M	99.47	93.2
Baidu [19]	1	10	1.3M	99.13	-
Center Face [40]	1	7	0.7M	99.28	94.9
Noisy Softmax [3]	1	8	WebFace+	99.18	94.88
Range loss [48]	1	28	1.5M	99.52	93.7
Augumentation [22]	1	19	1.5M	98.06	-
Center invariant loss [42]	1	22	WebFace	99.12	93.88
Feature transfer [45]	1	-	4.8M	99.37	-
Softmax Loss	1	64	WebFace	97.88	93.1
Softmax+Contrastive [29]	1	64	WebFace	98.78	93.5
Triplet Loss [27]	1	64	WebFace	98.70	93.4
L-Softmax Loss [21]	1	64	WebFace	99.10	94.0
Softmax+Center Loss [40]	1	64	WebFace	99.05	94.4
SphereFace (A-Softmax) [20]	1	64	WebFace	99.42	95.0
CosFace (LMCL) [36]	1	64	WebFace	99.33	96.1
Ours (Fair Loss)	1	50	WebFace	**99.57**	**96.2**

Table 3. Face verification accuracy (%) on LFW and YTF. "#Nets" represents the number of networks for ensemble and "#Layers" represents the number of layers in a network.

4.5. Experiments on MegaFace Challenge

MegaFace [14, 24] is a testing benchmark with very challenging tasks, for the purpose of evaluating face recognition methods at the million scale of distractors. The MegaFace datasets include a gallery set with more than 1 million face images collected from Yahoo's 100M Flickr set [34] and a probe set with 100K photos of 530 celebrities from FaceScrub [25]. In addition, MegaFace has two separate testing scenarios including identification and verification under two protocols (small or large training dataset). The training dataset is viewed as small if it has less than 0.5M images, large for other cases. For verification, we compare the performance on TAR under 10^{-6} FAR, where TAR and FAR denote True Accept Rate and False Accept Rate, respectively. We evaluate the performance of our method on both MegaFace Challenge 1 and Challenge 2.

MegaFace Challenge 1 (MF1). On MegaFace Challenge 1 [14], the gallery set includes 1 million photos that capture more than 690K different persons. We evaluate fair loss under the small training set protocol by training on CASIA-WebFace [44]. The results are shown in Table 4. Our method wins the first place of the verification test. For rank-1 identification, fair loss outperforms all the classical metric learning loss and large-margin loss with the same training dataset. Fair loss shows its superiority on learning discriminative features from imbalanced dataset.

MegaFace Challenge 2 (MF2). As for MegaFace Challenge 2 [24], all the algorithms are required to train on the dataset provided by MegaFace. The MF2 training dataset contains 4.7 million faces and 672K unique identities, which is a large-scale imbalanced dataset. The gallery set is different from MF1, incorporating 1M distractor images that are disjoint from MF2 training dataset. We compare our method with other advanced models in Table 5. Our fair loss

Method	Protocol	MF1 Rank1	MF1 Veri.
SIAT_MMLAB [40]	Small	65.23	76.72
DeepSense - Small	Small	70.98	82.85
Beijing FaceALL V2	Small	76.66	77.60
GRCCV	Small	**77.67**	74.88
Softmax Loss	Small	54.85	65.92
Softmax+Contrastive [29]	Small	65.21	78.86
Triplet Loss [27]	Small	64.79	78.32
L-Softmax Loss [21]	Small	67.12	80.42
Softmax+Center Loss [40]	Small	65.49	80.14
SphereFace (A-Softmax) [20]	Small	72.72	85.56
CosFace (LMCL) [36]	Small	77.11	89.88
Ours (Fair Loss)	Small	**77.45**	**92.87**

Table 4. Face identification and verification evaluation on MF1. "Rank1" refers to rank-1 identification accuracy with 1M distractors, and "Veri." refers to verification TAR for 10^{-6} FAR. The methods in the second cell and ours in the last cell use the same training dataset (CASIA-WebFace [44]).

Method	Protocol	MF2 Rank1	MF2 Veri.
3DiVi	Large	57.04	66.45
Team 2009	Large	58.93	71.12
NEC	Large	62.12	66.84
GRCCV	Large	75.77	74.84
SphereFace (A-Softmax) [20]	Large	71.17	84.22
CosFace (LMCL) [36]	Large	74.11	86.77
Ours (Fair Loss)	Large	**79.41**	**89.62**

Table 5. Face identification and verification evaluation on MF2. "Rank1" refers to rank-1 identification accuracy with 1M distractors. "Veri." refers to verification TAR for 10^{-6} FAR.

makes considerably better performance than the other models, which confirms the effectiveness of fair loss on large-scale imbalanced dataset. According to the leaderboard of MegaFace Challenge 2 [24], our method sets a new state-of-the-art on rank-1 identification and achieves runner-up performance on verification, with a single model to be tested. Note that most of the previous high performance models, including the existing state-of-the-art method, used model ensemble technique.

5. Conclusions

In this paper, we contribute to the improvement of deep face recognition with class imbalance problem by proposing a new loss function, namely fair loss, with adaptive margins. We use reinforcement learning to learn the margin adaptive strategy. Our approach by strategy based on CosFace obtains competitive results on several popular face benchmarks, which shows significant superiority on learning discriminative features from imbalanced datasets. Furthermore, we empirically demonstrate that the way to learn margin adaptive strategy by reinforcement learning can be used in various large-margin loss functions to make better performance in the case of class imbalance.

6. Acknowledgments

This work was supported by Canon Information Technology (Beijing) Co., Ltd. under Grant No. OLA18001.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, 2016.

[2] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority oversampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[3] Binghui Chen, Weihong Deng, and Junping Du. Noisy softmax: Improving the generalization ability of dcnn via postponing the early softmax saturation. In *CVPR*, 2017.

[4] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv:1512.01274*, 2015.

[5] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *CVPR*, 2019.

[6] Chris Drummond, Robert C Holte, et al. C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In *Workshop on Learning from Imbalanced Datasets II*, 2003.

[7] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *ICIC*, 2005.

[8] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *IJCNN*, 2008.

[9] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge & Data Engineering*, (9):1263–1284, 2008.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[11] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018.

[12] Chen Huang, Simon Lucey, and Deva Ramanan. Learning policies for adaptive tracking with deep feature cascades. In *ICCV*, 2017.

[13] Gary B Huang, Marwan Mattar, Tamara Berg, and Eric Learned-Miller. Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. Technical report, 2007.

[14] Ira Kemelmacher-Shlizerman, Steven M Seitz, Daniel Miller, and Evan Brossard. The megaface benchmark: 1 million faces for recognition at scale. In *CVPR*, 2016.

[15] Salman H Khan, Munawar Hayat, Mohammed Bennamoun, Ferdous A Sohel, and Roberto Togneri. Cost-sensitive learning of deep feature representations from imbalanced data. *IEEE Transactions on Neural Networks and Learning Systems*, 29(8):3573–3587, 2018.

[16] Bartosz Krawczyk, Michał Woźniak, and Gerald Schaefer. Cost-sensitive decision tree ensembles for effective imbalanced classification. *Applied Soft Computing*, 14:554–562, 2014.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[18] Shan Li and Weihong Deng. Deep facial expression recognition: A survey. *arXiv:1804.08348*, 2018.

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

[20] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In *CVPR*, 2017.

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

[22] Iacopo Masi, Anh Tun Trn, Tal Hassner, Jatuporn Toy Leksut, and Gérard Medioni. Do we really need to collect millions of faces for effective face recognition? In *ECCV*, 2016.

[23] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[24] Aaron Nech and Ira Kemelmacher-Shlizerman. Level playing field for million scale face recognition. In *CVPR*, 2017.

[25] Hong-Wei Ng and Stefan Winkler. A data-driven approach to cleaning large face datasets. In *ICIP*, 2014.

[26] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, et al. Deep face recognition. In *BMVC*, 2015.

[27] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 2015.

[28] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014.

[29] Yi Sun, Yuheng Chen, Xiaogang Wang, and Xiaoou Tang. Deep learning face representation by joint identification-verification. In *NIPS*, 2014.

[30] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deeply learned face representations are sparse, selective, and robust. In *CVPR*, 2015.

[31] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[32] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *CVPR*, 2014.

[33] Yuchun Tang, Yan-Qing Zhang, Nitesh V Chawla, and Sven Krasser. Svms modeling for highly imbalanced classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(1):281–288, 2009.

[34] Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. Yfcc100m: The new data in multimedia research. *arXiv:1503.01817*, 2015.

[35] Kai Ming Ting. A comparative study of cost-sensitive boosting algorithms. In *ICML*, 2000.

[36] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. In *CVPR*, 2018.

[37] Mei Wang and Weihong Deng. Deep face recognition: A survey. *arXiv:1804.06655*, 2018.

[38] Shoujin Wang, Wei Liu, Jia Wu, Longbing Cao, Qinxue Meng, and Paul J Kennedy. Training deep neural networks on imbalanced data sets. In *IJCNN*, 2016.

[39] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Learning to model the tail. In *NIPS*, 2017.

[40] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In *ECCV*, 2016.

[41] Lior Wolf, Tal Hassner, and Itay Maoz. Face recognition in unconstrained videos with matched background similarity. In *CVPR*, 2011.

[42] Yue Wu, Hongfu Liu, Jun Li, and Yun Fu. Deep face recognition with center invariant loss. In *Thematic Workshop of ACM-MM*, 2017.

[43] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.

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

[45] Xi Yin, Xiang Yu, Kihyuk Sohn, Xiaoming Liu, and Manmohan Chandraker. Feature transfer learning for deep face recognition with long-tail data. *arXiv:1803.09014*, 2018.

[46] Jianfu Zhang, Naiyan Wang, and Liqing Zhang. Multi-shot pedestrian re-identification via sequential decision making. In *CVPR*, 2018.

[47] 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.

[48] Xiao Zhang, Zhiyuan Fang, Yandong Wen, Zhifeng Li, and Yu Qiao. Range loss for deep face recognition with long-tailed training data. In *CVPR*, 2017.

[49] Zhi-Hua Zhou and Xu-Ying Liu. Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Transactions on Knowledge and Data Engineering*, 18(1):63–77, 2006.

[50] Zhi-Hua Zhou and Xu-Ying Liu. On multi-class cost-sensitive learning. *Computational Intelligence*, 26(3):232–257, 2010.