AdaCos: Adaptively Scaling Cosine Logits for Effectively Learning Deep Face Representations

Xiao Zhang[1]   Rui Zhao[2]   Yu Qiao[3]   Xiaogang Wang[1]   Hongsheng Li[1]

[1]CUHK-SenseTime Joint Laboratory, The Chinese University of Hong Kong   [2]SenseTime Research
[3]SIAT-SenseTime Joint Lab, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences

zhangx9411@gmail.com   zhaorui@sensetime.com   yu.qiao@siat.ac.cn   {xgwang, hsli}@ee.cuhk.edu.hk

Abstract

The cosine-based softmax losses [21, 28, 39, 8] and their variants [40, 38, 7] achieve great success in deep learning based face recognition. However, hyperparameter settings in these losses have significant influences on the optimization path as well as the final recognition performance. Manually tuning those hyperparameters heavily relies on user experience and requires many training tricks.

In this paper, we investigate in depth the effects of two important hyperparameters of cosine-based softmax losses, the scale parameter and angular margin parameter, by analyzing how they modulate the predicted classification probability. Based on these analysis, we propose a novel cosine-based softmax loss, AdaCos, which is hyperparameter-free and leverages an adaptive scale parameter to automatically strengthen the training supervisions during the training process. We apply the proposed AdaCos loss to large-scale face verification and identification datasets, including LFW [13], MegaFace [16], and IJB-C [23] 1:1 Verification. Our results show that training deep neural networks with the AdaCos loss is stable and able to achieve high face recognition accuracy. Our method outperforms state-of-the-art softmax losses [28, 40, 7] on all the three datasets.

1. Introduction

Recent years witnessed the breakthrough of deep Convolutional Neural Networks (CNNs) [17, 12, 25, 35] on significantly improving the performance of one-to-one $(1:1)$ face verification and one-to-many $(1:N)$ face identification tasks. The successes of deep face CNNs can be mainly credited to three factors: enormous training data [9], deep neural network architectures [10, 33] and effective loss functions [28, 21, 7]. Modern face datasets, such as LFW [13], CASIA-WebFace [43], MS1M [9] and MegaFace [24, 16], contain huge number of identities which enable the training of deep networks. A number of recent studies, such as DeepFace [36], DeepID2 [31], DeepID3 [32], VGGFace [25] and FaceNet [29], demon-

strated that properly designed network architectures also lead to improved performance.

Apart from the large-scale training data and deep structures, training losses also play key roles in learning accurate face recognition models [41, 6, 11]. Unlike image classification tasks, face recognition is essentially an open set recognition problem, where the testing categories (identities) are generally different from those used in training. To handle this challenge, most deep learning based face recognition approaches [31, 32, 36] utilize CNNs to extract feature representations from facial images, and adopt a metric (usually the cosine distance) to estimate the similarities between pairs of faces during inference.

However, such inference evaluation metric is not well considered in the methods with softmax cross-entropy loss function[1], which train the networks with the softmax loss but perform inference using cosine-similarities. To mitigate the gap between training and testing, recent works [21, 28, 39, 8] directly optimized cosine-based softmax losses. Moreover, angular margin-based terms [19, 18, 40, 38, 7] are usually integrated into cosine-based losses to maximize the angular margins between different identities. These methods improve the face recognition performance in the open-set setup. In spite of their successes, the training processes of cosine-based losses (and their variants introducing margins) are usually tricky and unstable. The convergence and performance highly depend on the hyperparameter settings of loss, which are determined empirically through large amount of trials. In addition, subtle changes of these hyperparameters may fail the entire training process.

In this paper, we investigate state-of-the-art cosine-based softmax losses [28, 40, 7], especially those aiming at maximizing angular margins, to understand how they provide supervisions for training deep neural networks. Each of the functions generally includes several hyperprameters, which have substantial impact on the final performance and are usually difficult to tune. One has to repeat training with dif-

---

[1]We denote it as "softmax loss" for short in the remaining sections.

ferent settings for multiple times to achieve optimal performance. Our analysis shows that different hyperparameters in those cosine-based losses actually have similar effects on controlling the samples' predicted class probabilities. Improper hyperparameter settings cause the loss functions to provide insufficient supervisions for optimizing networks.

Based on the above observation, we propose an adaptive cosine-based loss function, AdaCos, which automatically tunes hyperparameters and generates more effective supervisions during training. The proposed AdaCos dynamically scales the cosine similarities between training samples and corresponding class center vectors (the fully-connection vector before softmax), making their predicted class probability meets the semantic meaning of these cosine similarities. Furthermore, AdaCos can be easily implemented using built-in functions from prevailing deep learning libraries [26, 1, 5, 15]. The proposed AdaCos loss leads to faster and more stable convergence for training without introducing additional computational overhead.

To demonstrate the effectiveness of the proposed AdaCos loss function, we evaluated it on several face benchmarks, including LFW face verification [13], MegaFace one-million identification [24] and IJB-C [23]. Our method outperforms state-of-the-art cosine-based losses on all these benchmarks.

2. Related Works

Cosine similarities for inference. For learning deep face representations, feature-normalized losses are commonly adopted to enhance the recognition accuracy. Coco loss [20, 21] and NormFace [39] studied the effect of normalization and proposed two strategies by reformulating softmax loss and metric learning. Similarly, Ranjan *et al.* in [28] also discussed this problem and applied normalization on learned feature vectors to restrict them lying on a hypersphere. Movrever, compared with these hard normalization, ring loss [45] came up with a soft feature normalization approach with convex formulations.

Margin-based softmax loss. Earlier, most face recognition approaches utilized metric-targeted loss functions, such as triplet [41] and contrastive loss [6], which utilize Euclidean distances to measure similarities between features. Taking advantages of these works, center loss [42] and range loss [44] were proposed to reduce intra-class variations via minimizing distances within each class [2]. Following this, researchers found that constraining margin in Euclidean space is insufficient to achieve optimal generalization. Then angular-margin based loss functions were proposed to tackle the problem. Angular constraints were integrated into the softmax loss function to improve the learned face representation by L-softmax [19] and A-softmax [18]. CosFace [40], AM-softmax [38] and ArcFace [7] directly maximized angular margins and employed simpler and more intuitive loss functions compared with afore-mentioned methods.

Automatic hyperparameter tuning. The performance of an algorithm highly depends on hyperparameter settings. Grid and random search [3] are the most widely used strategies. For more automatic tuning, sequential model-based global optimization [14] is the mainstream choice. Typically, it performs inference with several hyperparameters settings, and chooses setting for the next round of testing based on the inference results. Bayesian optimization [30] and tree-structured parzen estimator approach [4] are two famous sequential model-based methods. However, these algorithms essentially run multiple trials to predict the optimized hyperparameter settings.

3. Investigation of hyperparameters in cosine-based softmax losses

In recent years, state-of-the-art cosine-based softmax losses, including L2-softmax [28], CosFace [40], ArcFace [7], significantly improve the performance of deep face recognition. However, the final performances of those losses are substantially affected by their hyperparameters settings, which are generally difficult to tune and require multiple trials in practice. We analyze two most important hyperparameters, the scaling parameter s and the margin parameter m, in cosine-based losses. Specially, we deeply study their effects on the prediction probabilities after softmax, which serves as supervision signals for updating entire neural network.

Let $\vec{x}_i$ denote the deep representation (feature) of the i-th face image of the current mini-batch with size N, and y_i be the corresponding label. The predicted classification probability $P_{i,j}$ of all N samples in the mini-batch can be estimated by the softmax function as

$$P_{i,j} = \frac{e^{f_{i,j}}}{\sum_{k=1}^{C} e^{f_{i,k}}}, \tag{1}$$

where $f_{i,j}$ is logit used as the input of softmax, $P_{i,j}$ represents its softmax-normalized probability of assigning $\vec{x}_i$ to class j, and C is the number of classes. The cross-entropy loss associated with current mini-batch is

$$\mathcal{L}_{\text{CE}} = -\frac{1}{N} \sum_{i=1}^{N} \log P_{i,y_i} = -\frac{1}{N} \sum_{i=1}^{N} \log \frac{e^{f_{i,y_i}}}{\sum_{k=1}^{C} e^{f_{i,k}}}. \tag{2}$$

Conventional softmax loss and state-of-the-art cosine-based softmax losses [28, 40, 7] calculate the logits $f_{i,j}$ in different ways. In conventional softmax loss, logits $f_{i,j}$ are obtained as the inner product between feature $\vec{x}_i$ and the j-th class weights $\vec{W}_j$ as $f_{i,j} = \vec{W}_j^{\mathrm{T}} \vec{x}_i$. In the cosine-based softmax losses [28, 40, 7], cosine similarity is calculated by $\cos\theta_{i,j} = \langle \vec{x}_i, \vec{W}_j \rangle / \|\vec{x}_i\| \|\vec{W}_j\|$. The logits $f_{i,j}$ are calculated as $f_{i,j} = s \cdot \cos\theta_{i,j}$, where s is a scale hyperparameter. To enforce angular margin on the representations, ArcFace [7] modified the loss to the form
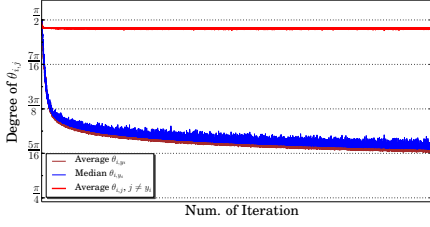
Figure 1: Changing process of angles in each mini-batch when training on WebFace. (Red) average angles in each mini-batch for non-corresponding classes, $\theta_{i,j}$ for $j \neq y_i$. (Blue) median angles in each mini-batch for corresponding classes, θ_{i,y_i}. (Brown) average angles in each mini-batch for corresponding classes, θ_{i,y_i}.

$$f_{i,j} = s \cdot \cos(\theta_{i,j} + \mathbb{1}\{j = y_i\} \cdot m), \qquad (3)$$

while CosFace [40] uses

$$f_{i,j} = s \cdot (\cos\theta_{i,j} - \mathbb{1}\{j = y_i\} \cdot m), \qquad (4)$$

where m is the margin. The indicator function $\mathbb{1}\{j = y_i\}$ returns 1 when $j = y_i$ and 0 otherwise. All margin-based variants decrease f_{i,y_i} associate with the correct class by subtracting margin m. Compared with the losses without margin, margin-based variants require f_{i,y_i} to be greater than other $f_{i,j}$ for $j \neq y_i$, by a specified m.

Intuitively, on one hand, the parameter s scales up the narrow range of cosine distances, making the logits more discriminative. On the other hand, the parameter m enlarges the margin between different classes to enhance classification ability. These hyperparameters eventually affect P_{i,y_i}. Empirically, an ideal hyperparameter setting should help $P_{i,j}$ to satisfy the following two properties: (1) Predicted probabilities P_{i,y_i} of each class (identity) should span to the range $[0, 1]$: the lower boundary of P_{i,y_i} should be near 0 while the upper boundary near 1; (2) Changing curve of P_{i,y_i} should have large absolute gradients around θ_{i,y_i} to make training effective.

3.1. Effects of the scale parameter s

The scale parameter s can significantly affect P_{i,y_i}. Intuitively, P_{i,y_i} should gradually increase from 0 to 1 as the angle θ_{i,y_i} decreases from $\frac{\pi}{2}$ to 0^2, *i.e.*, the smaller the angle between $\vec{x}_i$ and its corresponding class weight $\vec{W}_{y_i}$ is, the larger the probability should be. Both improper probability range and probability curves w.r.t. θ_{i,y_i} would negatively affect the training process and thus the recognition performance.

We first study the range of classification probability $P_{i,j}$. Given scale parameter s, the range of probabilities in all cosine-based softmax losses is

---

²Mathematically, θ can be any value in $[0, \pi]$. We empirically found, however, the maximum θ is always around $\frac{\pi}{2}$. See the red curve in Fig. 1 for examples.

$$\frac{1}{1 + (C-1) \cdot e^s} \leq P_{i,j} \leq \frac{e^s}{e^s + (C-1)}, \qquad (5)$$

where the lower boundary is achieved when $f_{i,j} = s \cdot 0 = 0$ and $f_{i,k} = s \cdot 1 = s$ for all $k \neq j$ in Eq. (1). Similarly, the upper bound is achieved when $f_{i,j} = s$ and $f_{i,k} = 0$ for all $k \neq j$. The range of $P_{i,j}$ approaches 1 when $s \to \infty$, *i.e.*,

$$\lim_{s \to +\infty} \left(\frac{e^s}{e^s + (C-1)} - \frac{1}{1 + (C-1) \cdot e^s} \right) = 1, \quad (6)$$

which means that the requirement of the range spanning $[0, 1]$ could be satisfied with a large s. However it does not mean that the larger the scale parameter, the better the selection is. In fact the probability range can easily approach a high value, such as 0.94 when class number $C = 10$ and scale parameter $s = 5.0$. But an oversized scale would lead to poor probability distribution, as will be discussed in the following paragraphs.

We investigate the influences of parameter s by taking P_{i,y_i} as a function of s and angle θ_{i,y_i} where y_i denotes the label of $\vec{x}_i$. Formally, we have

$$P_{i,y_i} = \frac{e^{f_{i,y_i}}}{e^{f_{i,y_i}} + B_i} = \frac{e^{s \cdot \cos\theta_{i,y_i}}}{e^{s \cdot \cos\theta_{i,y_i}} + B_i}, \qquad (7)$$

where $B_i = \sum_{k \neq y_i} e^{f_{i,k}} = \sum_{k \neq y_i} e^{s \cdot \cos\theta_{i,k}}$ are the logits summation of all non-corresponding classes for feature $\vec{x}_i$. We observe that the values of B_i are almost unchanged during the training process. This is because the angles $\theta_{i,k}$ for non-corresponding classes $k \neq y_i$ always stay around $\frac{\pi}{2}$ during training (see red curve in Fig. 1).

Therefore, we can assume B_i is constant, *i.e.*, $B_i \approx \sum_{k \neq y_i} e^{s \cdot \cos(\pi/2)} = C - 1$. We then plot curves of probabilities P_{i,y_i} w.r.t. θ_{i,y_i} under different setting of parameter s in Fig. 2(a). It is obvious that when s is too small (*e.g.*, $s = 10$ for class/identity number $C = 2,000$ and $C = 20,000$), the maximal value of P_{i,y_i} could not reach 1. This is undesirable because even when the network is very confident on a sample $\vec{x}_i$'s corresponding class label y_i, e.g. $\theta_{i,y_i} = 0$, the loss function would still penalize the classification results and update the network.

On the other hand, when s is too large (*e.g.*, $s = 64$), the probability curve P_{i,y_i} w.r.t. θ_{i,y_i} is also problematic. It would output a very high probability even when θ_{i,y_i} is close to $\pi/2$, which means that the loss function with large s may fail to penalize mis-classified samples and cannot effectively update the networks to correct mistakes.

In summary, the scaling parameter s has substantial influences to the range as well as the curves of the probabilities P_{i,y_i}, which are crucial for effectively training the deep network.

3.2. Effects of the margin parameter m

In this section, we investigate the effect of margin parameters m in cosine-based softmax losses (Eqs. (3) & (4)),
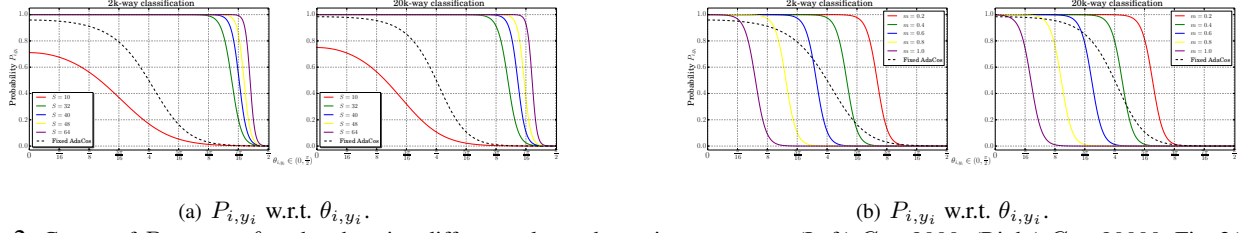
(a) P_{i,y_i} w.r.t. θ_{i,y_i}.

(b) P_{i,y_i} w.r.t. θ_{i,y_i}.

Figure 2: Curves of P_{i,y_i} w.r.t. θ_{i,y_i} by choosing different sclae and margin parameters. (Left) $C = 2000$. (Right) $C = 20000$. Fig. 2(a) is for choosing different scale parameters and Fig. 2(b) is for fixing $s = 30$ and choosing different margin parameters.

and their effects on feature $\vec{x}_i$'s predicted class probability P_{i,y_i}. For simplicity, we here study the margin parameter m for ArcFace (Eq. 3); while the similar conclusions also apply to the parameter m in CosFace (Eq. (4)).

We first re-write classification probability P_{i,y_i} following Eq. (7) as

$$P_{i,y_i} = \frac{e^{f_{i,y_i}}}{e^{f_{i,y_i}} + B_i} = \frac{e^{s \cdot \cos(\theta_{i,y_i} + m)}}{e^{s \cdot \cos(\theta_{i,y_i} + m)} + B_i}. \quad (8)$$

To study the influence of parameter m on the probability P_{i,y_i}, we assume both s and B_i are fixed. Following the discussion in Section 3.1, we set $B_i \approx C - 1$, and fix $s = 30$. The probability curves P_{i,y_i} w.r.t. θ_{i,y_i} under different m are shown in Fig. 2(b).

According to Fig. 2(b), increasing the margin parameter shifts probability P_{i,y_i} curves to the left. Thus, with the same θ_{i,y_i}, larger margin parameters lead to lower probabilities P_{i,y_i} and thus larger loss even with small angles θ_{i,y_i}. In other words, the angles θ_{i,y_i} between the feature $\vec{x}_i$ and its corresponding class's weights $\vec{W}_{y_i}$ have to be very small for sample i being correctly classified. This is the reason why margin-based losses provide stronger supervisions for the same θ_{i,y_i} than conventional cosine-based losses. Proper margin settings have shown to boost the final recognition performance in [40, 7].

Although larger margin m provides stronger supervisions, it should not be too large either. When m is oversized (*e.g.*, $m = 1.0$), the probabilities P_{i,y_i} becomes unreliable. It would output probabilities around 0 even θ_{i,y_i} is very small. This lead to large loss for almost all samples even with very small sample-to-class angles, which makes the training difficult to converge. In previous methods, the margin parameter selection is an ad-hoc procedure and has no theoretical guidance for most cases.

3.3. Summary of the hyparameter study

According to our analysis, we can draw the following conclusions:

(1) Hyperparameters scale s and margin m can substantially influence the prediction probability P_{i,y_i} of feature $\vec{x}_i$ with ground-truth identity/category y_i. For the scale parameter s, too small s would limit the maximal value of P_{i,y_i}. On the other hand, too large s would make most predicted probabilities P_{i,y_i} to be 1, which makes the training loss in-

sensitive to the correctness of θ_{i,y_i}. For the margin parameter m, a too small margin is not strong enough to regularize the final angular margin, while an oversized margin makes the training difficult to converge.

(2) The effect of scale s and margin m can be unified to modulate the mapping from cosine distances $\cos \theta_{i,y_i}$ to the prediction probability P_{i,y_i}. As shown in Fig. 2(a) and Fig. 2(b), both small scales and large margins have similar effect on θ_{i,y_i} for strengthening the supervisions, while both large scales and small margins weaken the supervisions. Therefore it is feasible and promising to control the probability P_{i,y_i} using one single hyperparameter, either s or m. Considering the fact that s is more related to the range of P_{i,y_i} that required to span $[0, 1]$, we will focus on automatically tuning the scale parameter s in the reminder of this paper.

4. The cosine-based softmax loss with adaptive scaling

Based on our previous studies on the hyperparameters of the cosine-based softmax loss functions, in this section, we propose a novel loss with a self-adaptive scaling scheme, namely AdaCos, which does not require the ad-hoc and time-consuming manual parameter tuning. Training with the proposed loss does not only facilitate convergence but also results in higher recognition accuracy.

Our previous studies on Fig. 1 show that during the training process, the angles $\theta_{i,k}$ for $k \neq y_i$ between the feature $\vec{x}_i$ and its non-corresponding weights $\vec{W}_{k \neq y_i}$ are almost always close to $\frac{\pi}{2}$, In other words, we could safely assume that $B_i \approx \sum_{k \neq y_i} e^{s \cdot \cos(\pi/2)} = C - 1$ in Eq. (7). Obviously, it is the probability P_{i,y_i} of feature x_i belonging to its corresponding class y_i that has the most influence on supervision for network training. Therefore, we focus on designing an adaptive scale parameter for controling the probabilities P_{i,y_i}.

From the curves of P_{i,y_i} w.r.t. θ_{i,y_i} (Fig. 2(a)), we observe that the scale parameter s does not only simply affect P_{i,y_i}'s boundary of of determining correct/incorrect but also squeezes/stretches the P_{i,y_i} curvature; In contrast to scale s, margin parameter m only shifts the curve in phase. We therefore propose to automatically tune the scale parameter s and eliminate the margin parameter m from our loss func-

tion, which makes our proposed AdaCos loss different from state-of-the-art softmax loss variants with angular margin. With softmax function, the predicted probability can be defined by

$$P_{i,j} = \frac{e^{\tilde{s}\cdot\cos\theta_{i,j}}}{\sum_{k=1}^{C} e^{\tilde{s}\cdot\cos\theta_{i,k}}}, \tag{9}$$

where $\tilde{s}$ is the automatically tuned scale parameter to be discussed below.

Let us first re-consider the P_{i,y_i} (Eq. (7)) as a function of θ_{i,y_i}. Note that θ_{i,y_i} represents the angle between sample $\vec{x}_i$ and the weight vector of its ground truth category y_i. For network training, we hope to minimize θ_{i,y_i} with the supervision from the loss function L_{CE}. Our objective is choose a suitable scale $\tilde{s}$ which makes predicted probability P_{i,y_i} change significantly with respect to θ_{i,y_i}. Mathematically, we find the point where the absolute gradient value $\|\frac{\partial P_{i,y_i}(\theta)}{\partial\theta}\|$ reaches its maximum, when the second-order derivative of P_{i,y_i} at θ_0 equals 0, *i.e.*,

$$\frac{\partial^2 P_{i,y_i}(\theta_0)}{\partial\theta_0{}^2} = 0, \tag{10}$$

where $\theta_0 \in [0, \frac{\pi}{2}]$. Combining Eqs. (7) and (10), we obtain the relation between the scale parameter s and the point $(\theta_0, P(\theta_0))$ as

$$s_0 = \frac{\log B_i}{\cos\theta_0}, \tag{11}$$

where B_i can be well approximated as $B_i = \sum_{k\neq y_i} e^{s\cdot\cos\theta_{i,k}} \approx C - 1$ since the angles $\theta_{i,k}$ distribute around $\pi/2$ during training (see Eq. (7) and Fig. 1). Then the task of automatically determining $\tilde{s}$ would reduce to select an reasonable central angle $\tilde{\theta}$ in $[0, \pi/2]$.

4.1. Automatically choosing a fixed scale parameter

Since $\frac{\pi}{4}$ is in the center of $[0, \frac{\pi}{2}]$, it is natural to regard $\pi/4$ as the point, *i.e.* setting $\theta_0 = \pi/4$ for figuring out an effective mapping from angle θ_{i,y_i} to the probability P_{i,y_i}. Then the supervisions determined by P_{i,y_i} would be back-propagated to update θ_{i,y_i} and further to update network parameters. According to Eq. (11), we can estimate the corresponding scale parameter s_f as

$$\tilde{s}_f = \frac{\log B_i}{\cos\frac{\pi}{4}} = \frac{\log\sum_{k\neq y_i} e^{s\cdot\cos\theta_{i,k}}}{\cos\frac{\pi}{4}} \tag{12}$$
$$\approx \sqrt{2}\cdot\log(C-1)$$

where B_i is approximated by $C - 1$.

For such an automatically-chosen fixed scale parameter $\tilde{s}_f$ (see Figs. 2(a) and 2(b)), it depends on the number of classes C in the training set and also provides a good guideline for existing cosine distance based softmax losses to choose their scale parameters. In contrast, the scaling parameters in existing methods was manually set according to human experience. It acts as a good baseline method for our dynamically tuned scale parameter $\tilde{s}_d$ in the next section.

4.2. Dynamically adaptive scale parameter

As Fig. 1 shows, the angles θ_{i,y_i} between features $\vec{x}_i$ and their ground-truth class weights $\vec{W}_{y_i}$ gradually decrease as the training iterations increase; while the angles between features $\vec{x}_i$ and non-corresponding classes $\vec{W}_{j\neq y_i}$ become stabilize around $\frac{\pi}{2}$, as shown in Fig. 1.

Although our previously fixed scale parameter $\tilde{s}_f$ behaves properly as θ_{i,y_i} changes over $[0, \frac{\pi}{2}]$, it does not take into account the fact that θ_{i,y_i} gradually decrease during training. Since smaller θ_{i,y_i} gains higher probability P_{i,y_i} and thus gradually receives weaker supervisions as the training proceeds, we therefore propose a dynamically adaptive scale parameter $\tilde{s}_d$ to gradually apply stricter requirement on the position of θ_0 which can progressively enhance the supervisions throughout the training process.

Formally we introduce a modulating indicator variable $\theta_{med}^{(t)}$, which is the median of all corresponding classes' angles, $\theta_{i,y_i}^{(t)}$, from the mini-batch of size N at the t-th iteration. $\theta_{med}^{(t)}$ roughly represents the current network's degree of optimization on the mini-batch. When the median angle is large, it denotes that the network parameters are far from optimum and less strict supervisions should be applied to make the training converge more stably; when the median angle $\theta_{med}^{(t)}$ is small, it denotes that the network is close to optimum and stricter supervisions should be applied to make the intra-class angles θ_{i,y_i} become even smaller. Based on this observation, we set the central angle $\tilde{\theta}_0^{(t)} = \theta_{med}^{(t)}$. We also introduce $B_{avg}^{(t)}$ as the average of $B_i^{(t)}$ as

$$B_{avg}^{(t)} = \frac{1}{N}\sum_{i\in\mathcal{N}^{(t)}} B_i^{(t)} = \frac{1}{N}\sum_{i\in\mathcal{N}^{(t)}}\sum_{k\neq y_i} e^{\tilde{s}_d^{(t-1)}\cdot\cos\theta_{i,k}}, \tag{13}$$

where $\mathcal{N}^{(t)}$ denotes the face identity indices in the mini-batch at the t-th iteration. Unlike approximating $B_i \approx C-1$ for the fixed adaptive scale parameter $\tilde{s}_f$, here we estimate $B_i^{(t)}$ using the scale parameter $\tilde{s}_d^{(t-1)}$ of previous iteration, which provides us a more accurate approximation. Be reminded that $B_i^{(t)}$ also includes dynamic scale $\tilde{s}_d^{(t)}$. We can obtain it by solving the nonlinear function given by the above equation. In practice, we notice that $\tilde{s}_d^{(t)}$ changes very little following iterations. So, we just use $\tilde{s}_d^{(t-1)}$ to calculate $B_i^{(t)}$ with Eq. (7). Then we can obtain dynamic scale $\tilde{s}_d^{(t)}$ directly with Eq. (11). So we have:

$$\tilde{s}_d^{(t)} = \frac{\log B_{avg}^{(t)}}{\cos\theta_{med}^{(t)}}, \tag{14}$$

where $B_{avg}^{(t)}$ is related to the dynamic scale parameter. We estimate it using the scale parameter $\tilde{s}_d^{(t-1)}$ of the previous iteration.

At the begin of the training process, the median angle $\theta^{(t)}_{\text{med}}$ of each mini-batch might be too large to impose enough supervisions for training. We therefore force the central angle $\theta^{(t)}_{\text{med}}$ to be less than $\frac{\pi}{4}$. Our dynamic scale parameter for the t-th iteration could then be formulated as

$$\tilde{s}^{(t)}_d = \begin{cases} \sqrt{2}\cdot\log{(C-1)} & t=0, \\ \dfrac{\log B^{(t)}_{\text{avg}}}{\cos\left(\min(\frac{\pi}{4},\theta^{(t)}_{\text{med}})\right)} & t\geq 1, \end{cases} \quad (15)$$

where $\tilde{s}^{(0)}_d$ is initialized as our fixed scale parameter $\tilde{s}_f$ when $t=0$.

Substituting $\tilde{s}^{(t)}_d$ into $f_{i,j}=\tilde{s}^{(t)}_d\cdot\cos\theta_{i,j}$, the corresponding gradients can be calculated as follows

$$\frac{\partial\mathcal{L}(\vec{x}_i)}{\partial\vec{x}_i} = \sum_{j=1}^{C}(P^{(t)}_{i,j}-\mathbb{1}(y_i=j)\cdot\tilde{s}^{(t)}_d\frac{\partial\cos\theta_{i,j}}{\partial\vec{x}_i},$$

$$\frac{\partial\mathcal{L}(\vec{W}_j)}{\partial\vec{W}_j} = (P^{(t)}_{i,j}-\mathbb{1}(y_i=j))\cdot\tilde{s}^{(t)}_d\frac{\partial\cos\theta_{i,j}}{\partial\vec{W}_j}, \quad (16)$$

where $\mathbb{1}$ is the indicator function and

$$P^{(t)}_{i,j} = \frac{e^{\tilde{s}^{(t)}_d\cdot\cos\theta_{i,j}}}{\sum_{k=1}^{C}e^{\tilde{s}^{(t)}_d\cdot\cos\theta_{i,k}}}. \quad (17)$$

Eq. (17) shows that the dynamically adaptive scale parameter $\tilde{s}^{(t)}_d$ influences classification probabilities differently at each iteration and also effectively affects the gradients (Eq. (16)) for updating network parameters. The benefit of dynamic AdaCos is that it can produce reasonable scale parameter by sensing the training convergence of the model in the current iteration.

5. Experiments

We examine the proposed AdaCos loss function on several public face recognition benchmarks and compare it with state-of-the-art cosine-based softmax losses. The compared losses include l2-softmax [28], CosFace [40], and ArcFace [7]. We present evaluation results on LFW [13], MegaFace 1-million Challenge [24], and IJB-C [23] data. We also present results on some exploratory experiments to show the convergence speed and robustness against low-resolution images.

Preprocessing. We use two public training datasets, CASIA-WebFace [43] and MS1M [9], to train CNN models with our proposed loss functions. We carefully clean the noisy and low-quality images from the datasets. The cleaned WebFace [43] and MS1M [9] contain about $0.45M$ and $2.35M$ facial images, respectively. All models are trained based on these training data and directly tested on the test splits of the three datasets. RSA [22] is applied to the images to extract facial areas. Then, according to detected facial landmarks, the faces are aligned through similarity transformation and resized to the size 144×144. All

Method	1st	2nd	3rd	Average Acc.
Softmax	93.05	92.92	93.27	93.08
l2-softmax [28]	98.22	98.27	98.08	98.19
CosFace [40]	99.37	99.35	99.42	99.38
ArcFace [7]	99.55	99.37	99.43	99.45
Fixed AdaCos	99.63	99.62	99.55	**99.60**
Dyna. AdaCos	99.73	99.72	99.68	**99.71**

Table 1: Recognition accuracy on LFW by ResNet-50 trained with different compared losses. All the methods are trained on the cleaned WebFace [43] training data and tested on LFW for three times to obtain the average accuracy.
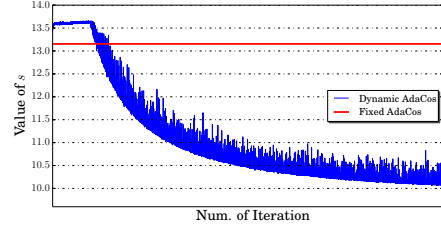

Figure 3: The change of the fixed adaptive scale parameter $\tilde{s}_f$ and dynamic adaptive scale parameter $\tilde{s}^{(t)}_d$ when training on the cleaned WebFace dataset. The dynamic scale parameter $\tilde{s}^{(t)}_d$ gradually and automatically decreases to strengthen training supervisions for feature angles θ_{i,y_i}, which validates our assumption on the adaptive scale parameter in our proposed AdaCos loss. Best viewed in color.

image pixel values are subtracted with the mean 127.5 and dividing by 128.

5.1. Results on LFW

The LFW [13] dataset collected thousands of identities from the inertnet. Its testing protocol contains about $13,000$ images for about $1,680$ identities with a total of $6,000$ ground-truth matches. Half of the matches are positive while the other half are negative ones. LFW's primary difficulties lie in face pose variations, color jittering, illumination variations and aging of persons. Note portion of the pose variations can be eliminated by the RSA [22] facial landmark detection and alignment algorithm, but there still exist some non-frontal facial images which can not be aligned by RSA [22] and then aligned manually.

5.1.1 Comparison on LFW

For all experiments on LFW [13], we train ResNet-50 models [10] with batch size of 512 on the cleaned WebFace [43] dataset. The input size of facial image is 144×144 and the feature dimension input into the loss function is 512. Different loss functions are compared with our proposed AdaCos losses.

Results in Table 1 show the recognition accuracies of models trained with different softmax loss functions. Our proposed AdaCos losses with fixed and dynamic scale parameters (denoted as Fixed AdaCos and Dyna. AdaCos)
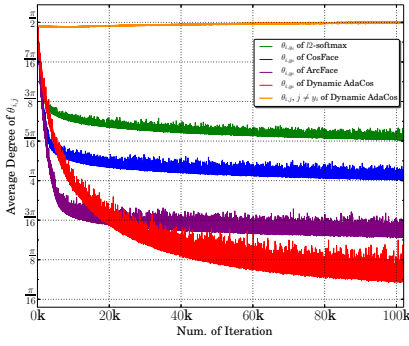
Figure 4: The change of θ_{i,y_i} when training on the cleaned Web-Face dataset. θ_{i,y_i} represents the angle between the feature vector of i-th sample and the weight vector of its ground truth category y_i. Curves calculated by proposed dynamic AdaCos loss, $l2$-softmax loss [28], CosFace [40] and ArcFace [7] are shown. Best viewed in color.

surpass the state-of-the-art cosine-based softmax losses under the same training configuration. For the hyperparameter settings of the compared losses, the scaling parameter is set as 30 for $l2$-softmax [28], CosFace [40] and Arc-Face [7]; the margin parameters are set as 0.25 and 0.5 for CosFace [40], and ArcFace [7], respectively. Since LFW is a relatively easy evaluation set, we train and test all losses for three times. The average accuracy of our proposed dynamic AdaCos is 0.26% higher than state-of-the-art Arc-Face [7] and 1.52% than $l2$-softmax [28].

5.1.2 Exploratory Experiments

The change of scale parameters and feature angles during training. In this part, we will show the change of scale parameter $\tilde{s}_d^{(t)}$ and feature angles $\theta_{i,j}$ during training with our proposed AdaCos loss. The scale parameter $\tilde{s}_d^{(t)}$ changes along with the current recognition performance of the model, which continuously strengthens the supervisions by gradually reducing θ_{i,y_i} and thus shrinking $\tilde{s}_d^{(t)}$. Fig. 3 shows the change of the scale parameter s with our proposed fixed AdaCos and dynamic AdaCos losses. For the dynamic AdaCos loss, the scale parameter $\tilde{s}_d^{(t)}$ adaptively decreases as the training iterations increase, which indicates that the loss function provides stricter supervisions to update network parameters. Fig. 4 illustrates the change of $\theta_{i,j}$ by our proposed dynamic AdaCos and $l2$-softmax. The average (orange curve) and median (green curve) of θ_{i,y_i}, which indicating the angle between a sample and its ground-truth category, gradually reduce while the average (maroon curve) of $\theta_{i,j}$ where $j \neq y_i$ remains nearly $\frac{\pi}{2}$. Compared with $l2$-softmax loss, our proposed loss could achieve much smaller sample feature to category angles on the ground-truth classes and leads to higher recognition accuracies.

Convergence rates. Convergence rate is an important

Method	Num. of Iteration			
	25k	50k	75k	100k
Softmax	70.15	85.33	89.50	93.05
$l2$-softmax [28]	79.08	88.52	93.38	98.22
CosFace [40]	78.17	90.87	98.52	99.37
ArcFace [7]	82.43	92.37	98.78	99.55
Fixed AdaCos	85.10	94.38	99.05	99.63
Dyna. AdaCos	**88.52**	**95.78**	**99.30**	**99.73**

Table 2: Convergence rates of different softmax losses. At the same iterations, training with our proposed dynamic AdaCos loss leads to the best recognition accuracy.
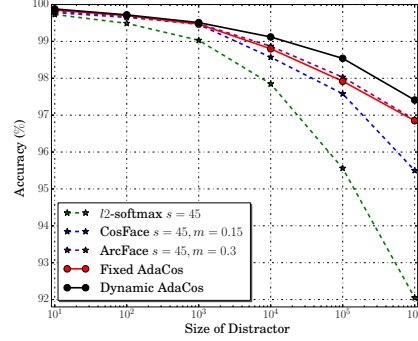

Figure 5: Recognition accuracy curves on MegaFace dataset by Inception-ResNet [34] models trained with different softmax losses and on the same cleaned WebFace [43] and MS1M [9] training data. Best viewed in color.

indicator of efficiency of loss functions. We examine the convergence rates of several cosine-based losses at different training iterations. The training configurations are same as Table 1. Results in Table 2 reveal that the convergence rates when training with the AdaCos losses are much higher.

5.2. Results on MegaFace

We then evaluate the performance of proposed AdaCos on the MegaFace Challenge [16], which is a publicly available identification benchmark, widely used to test the performance of facial recognition algorithms. The gallery set of MegaFace incorporates over 1 million images from 690K identities collected from Flickr photos [37]. We follow Arc-Face [7]'s testing protocol, which cleaned the dataset to make the results more reliable. We train the same Inception-ResNet [33] models with CASIA-WebFace [43] and MS1M [9] training data, where overlapped subjects are removed.

Table 3 and Fig. 5 summarize the results of models trained on both WebFace and MS1M datasets and tested on the cleaned MegaFace dataset. The proposed AdaCos and state-of-the-art softmax losses are compared, where the dynamic AdaCos loss outperforms all compared losses on the MegaFace.

5.3. Results on IJB-C 1:1 verification protocol

The IJB-C dataset [23] contains about $3,500$ identities with a total of $31,334$ still facial images and $117,542$ unconstrained video frames. In the 1:1 verification, there are

Method	Size of MegaFace Distractor					
	10^1	10^2	10^3	10^4	10^5	10^6
$l2$-softmax	99.73%	99.49%	99.03%	97.85%	95.56%	92.05%
CosFace	99.82%	99.68%	99.46%	98.57%	97.58%	95.50%
ArcFace	99.78%	99.65%	99.48%	98.87%	98.03%	96.88%
Fixed AdaCos	99.85%	99.70%	99.47%	98.80%	97.92%	96.85%
Dynamic AdaCos	**99.88%**	**99.72%**	**99.51%**	**99.02%**	**98.54%**	**97.41%**

Table 3: Recognition accuracy on MegaFace by Inception-ResNet [34] models trained with different compared softmax loss and the same cleaned WebFace [43] and MS1M [9] training data.

Method	True Accept Rate @ False Accept Rate						
	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}
FaceNet [29]	92.45%	81.71%	66.45%	48.69%	33.30%	20.95%	-
VGGFace [25]	95.64%	87.13%	74.79%	59.75%	43.69%	32.20%	-
Crystal Loss [27]	99.06%	97.66%	95.63%	92.29%	87.35%	81.15%	71.37%
$l2$-softmax	98.40%	96.45%	92.78%	86.33%	77.25%	62.61%	26.67%
CosFace [40]	99.01%	97.55%	95.37%	91.82%	86.94%	76.25%	61.72%
ArcFace [7]	**99.07%**	**97.75%**	95.55%	92.13%	87.28%	82.15%	72.28%
Fixed AdaCos	99.05%	97.70%	95.48%	92.35%	87.87%	82.38%	72.66%
Dynamic AdaCos	99.06%	97.72%	**95.65%**	**92.40%**	**88.03%**	**83.28%**	**74.07%**

Table 4: True accept rates by different compared softmax losses on the IJB-C 1:1 verification task. The same training data (WebFace [43] and MS1M [9]) and Inception-ResNet [33] networks are used. The results of FaceNet [29], VGGFace [25], and Crystal Loss [27] are from [27].
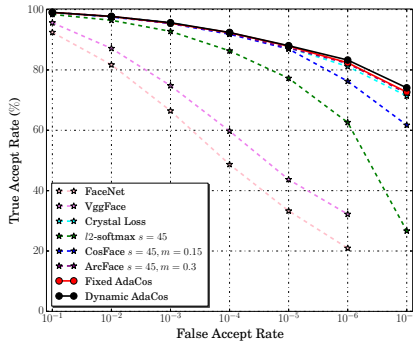

Figure 6: TARs by different compared softmax losses on the IJB-C 1:1 verification task. The same training data (WebFace [43] and MS1M [9]) and Inception-ResNet [33] are used. The results of FaceNet [29], VGGFace [25] are reported in Crystal Loss [27].

$19,557$ positive matches and $15,638,932$ negative matches, which allow us to evaluate TARs at various FARs (*e.g.*, 10^{-7}).

We compare the softmax loss functoins, including the proposed AdaCos, $l2$-softmax [28], CosFace [40], and ArcFace [7] with the same training data (WebFace [43] and MS1M [9]) and network architecture (Inception-ResNet [33]). We also report the results of FaceNet [29], VGGFace [36] listed in Crystal loss [27]. Table 4 and Fig. 6 exhibit their performances on the IJB-C 1:1 verification. Our proposed dynamic AdaCos achieves the best performance.

6. Conclusions

In this work, we argue that the bottleneck of existing cosine-based softmax losses may primarily comes from the mis-match between cosine distance $\cos\theta_{i,y_i}$ and the clas-sification probability P_{i,y_i}, which limits the final recognition performance. To address this issue, we first deeply analyze the effects of hyperparameters in cosine-based softmax losses from the perspective of probability. Based on these analysis, we propose the AdaCos which automatically adjusts an adaptive parameter $\tilde{s}_d^{(t)}$ in order to reformulate the mapping between cosine distance and classification probability. Our proposed AdaCos loss is simple yet effective. We demonstrate its effectiveness and efficiency by exploratory experiments and report its state-of-the-art performances on several public benchmarks.

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*, volume 16, pages 265–283, 2016.

[2] Peter N. Belhumeur, João P Hespanha, and David J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on pattern analysis and machine intelligence*, 19(7):711–720, 1997.

[3] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

[4] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems*, pages 2546–2554, 2011.

[5] 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 preprint arXiv:1512.01274*, 2015.

[6] 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.

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

[8] Siddharth Gopal and Yiming Yang. Von mises-fisher clustering models. In *International Conference on Machine Learning*, pages 154–162, 2014.

[9] 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.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

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

[12] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. *arXiv preprint arXiv:1709.01507*, 2017.

[13] 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.

[14] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *International Conference on Learning and Intelligent Optimization*, pages 507–523. Springer, 2011.

[15] 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*, pages 675–678. ACM, 2014.

[16] 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.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[18] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, 2017.

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

[20] Yu Liu, Hongyang Li, and Xiaogang Wang. Learning deep features via congenerous cosine loss for person recognition. *arXiv preprint arXiv:1702.06890*, 2017.

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

[22] Yu Liu, Hongyang Li, Junjie Yan, Fangyin Wei, Xiaogang Wang, and Xiaoou Tang. Recurrent scale approximation for object detection in cnn. In *IEEE International Conference on Computer Vision*, 2017.

[23] Brianna Maze, Jocelyn Adams, James A Duncan, Nathan Kalka, Tim Miller, Charles Otto, Anil K Jain, W Tyler Niggel, Janet Anderson, Jordan Cheney, et al. Iarpa janus benchmark–c: Face dataset and protocol. In *11th IAPR International Conference on Biometrics*, 2018.

[24] 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.

[25] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition.

[26] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[27] Rajeev Ranjan, Ankan Bansal, Hongyu Xu, Swami Sankaranarayanan, Jun-Cheng Chen, Carlos D Castillo, and Rama Chellappa. Crystal loss and quality pooling for unconstrained face verification and recognition. *arXiv preprint arXiv:1804.01159*, 2018.

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

[29] 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.

[30] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[31] 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.

[32] Yi Sun, Ding Liang, Xiaogang Wang, and Xiaoou Tang. Deepid3: Face recognition with very deep neural networks. *arXiv preprint arXiv:1502.00873*, 2015.

[33] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the

impact of residual connections on learning. In *AAAI*, volume 4, page 12, 2017.

[34] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, page 12, 2017.

[35] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[36] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1701–1708, 2014.

[37] 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 preprint arXiv:1503.01817*, 2015.

[38] Feng Wang, Weiyang Liu, Haijun Liu, and Jian Cheng. Additive margin softmax for face verification. *arXiv preprint arXiv:1801.05599*, 2018.

[39] Feng Wang, Xiang Xiang, Jian Cheng, and Alan L Yuille. Normface: l_2 hypersphere embedding for face verification. *arXiv preprint arXiv:1704.06369*, 2017.

[40] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Zhifeng Li, Dihong Gong, Jingchao Zhou, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. *arXiv preprint arXiv:1801.09414*, 2018.

[41] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb):207–244, 2009.

[42] 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.

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

[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.