SphereFace Revived: Unifying Hyperspherical Face Recognition

Weiyang Liu*, Yandong Wen*, Bhiksha Raj, *Fellow, IEEE*, Rita Singh, Adrian Weller

Abstract—This paper addresses the deep face recognition problem under an open-set protocol, where ideal face features are expected to have smaller maximal intra-class distance than minimal inter-class distance under a suitably chosen metric space. To this end, hyperspherical face recognition, as a promising line of research, has attracted increasing attention and gradually become a major focus in face recognition research. As one of the earliest works in hyperspherical face recognition, SphereFace explicitly proposed to learn face embeddings with large inter-class angular margin. However, SphereFace still suffers from severe training instability which limits its application in practice. In order to address this problem, we introduce a unified framework to understand large angular margin in hyperspherical face recognition. Under this framework, we extend the study of SphereFace and propose an improved variant with substantially better training stability – SphereFace-R. Specifically, we propose two novel ways to implement the multiplicative margin, and study SphereFace-R under three different feature normalization schemes (no feature normalization, hard feature normalization and soft feature normalization). We also propose an implementation strategy – "characteristic gradient detachment" – to stabilize training. Extensive experiments on SphereFace-R show that it is consistently better than or competitive with state-of-the-art methods.

Index Terms—Hypersphere, face recognition, angular margin, loss function

1 Introduction

Recent years have witnessed the tremendous success of deep face recognition (FR). Owing to the rapid development in discriminative loss functions [1], [2], [3], [4], [5] that promote large inter-class feature margin, the performance of deep FR has dramatically improved. These loss functions share a common goal to project deeply learned face embeddings onto a hypersphere and incorporate large geodesic inter-class margins. We call this series of deep FR methods *hyperspherical face recognition*.

Previous deep FR methods [6], [7] typically train neural networks by classifying identities in a training set. Such a training target largely deviates from the open-set testing (*i.e.*, to determine whether two face images belong to the same person) in two aspects: *(i)* similarity measure differs in training and testing; *(ii)* open-set testing must solve a metric learning problem [2] where the goal is to learn large-margin features, while training aims to solve a closed-set classification problem where the goal is to learn separable features. Motivated by the mismatch between training and testing in deep FR, hyperspherical FR aims to bridge the gap by *(i)* constraining the face embeddings on the hypersphere (*i.e.*, using cosine similarity for both training and testing), and *(ii)* incorporating large geodesic margin on the hypersphere. Another motivation for hyperspherical FR comes from the observation that deep features are intrinsically discriminative on a hypersphere [8]. Hyperspherical FR essentially focuses on answering the question: *How to effectively and stably incorporate large angular margin to face embeddings?*

Large-margin softmax [1] is one of the first methods to incorporate large angular margin to deeply learned features. The core idea is to use a monotonically decreasing lower bound function $\psi(\theta_{(\boldsymbol{x},\boldsymbol{W}_y)})$ to replace the target angular activation $\cos(\theta_{(\boldsymbol{x},\boldsymbol{W}_y)})$ in the softmax-based loss function, where $\theta_{(\boldsymbol{x},\boldsymbol{W}_y)}$ denotes the angle between deep feature $\boldsymbol{x}$ and the classifier of the target class $\boldsymbol{W}_y$ (y is the label of $\boldsymbol{x}$). The intuition is that the function $\psi(\theta_{(\boldsymbol{x},\boldsymbol{W}_y)})$ will make the angle $\theta_{(\boldsymbol{x},\boldsymbol{W}_y)}$ smaller in order to achieve the same value of $\cos(\theta_{(\boldsymbol{x},\boldsymbol{W}_y)})$. Such a design will encourage the deep features to have large inter-class margins on the unit hypersphere. Most popular hyperspherical FR methods [2], [3], [4], [5] adopt this design principle.

Built upon [1], SphereFace [2] takes one step further by explicitly constraining decision boundaries on the hypersphere and simultaneously incorporating angular margins. Inspired by SphereFace, there is a series of work [3], [4], [5] that design alternative lower bound target function $\psi(\theta_{(\boldsymbol{x},\boldsymbol{W}_y)})$ to achieve angular margin. Based on how $\psi(\theta_{(\boldsymbol{x},\boldsymbol{W}_y)})$ is constructed, loss functions in hyperspherical FR can be divided into *additive margin* [3], [4], [5] and *multiplicative margin* [1], [2]. As a representative multiplicative margin method, SphereFace renders promising geometric insights. However, in contrast to additive margin, SphereFace is known to be highly non-trivial to train, typically requiring a number of bells and whistles to stabilize its training, which limits its potential application.

In order to address this shortcoming, we take a detour by first identifying an intrinsic connection that bridges different margin designs [1], [2], [3], [4], [5], [9] in hyperspherical FR. We formulate this connection with a unified large-margin framework for hyperspherical FR. In this framework, we

- *W. Liu and A. Weller are with the Department of Engineering, University of Cambridge, United Kingdom. W. Liu is also with the Max Planck Institute for Intelligent Systems, Tübingen, Germany. A. Weller is also with The Alan Turing Institute, London, United Kingdom. E-mail: wl396@cam.ac.uk, aw665@cam.ac.uk*
- *Y. Wen, B. Raj and R. Singh are with the Department of Electrical and Computer Engineering, Carnegie Mellon University, United States. E-mail: yandongw@andrew.cmu.edu, {bhiksha,rsingh}@cs.cmu.edu*
- **W. Liu and Y. Wen have contributed equally to this work.*

Accepted for publication on 1 March 2022

summarize a general principle for any loss function to achieve large angular margin. Following this principle, most existing hyperspherical FR methods can be viewed as special instantiations. This framework helps us gain a deeper understanding of hyperspherical FR, and serves as a portal to design new loss functions.

Under this unified framework, we extend our previous study of SphereFace [2] by proposing alternative yet effective ways to implement the multiplicative margin with improved training stability and better empirical performance. Specifically, the original realization of multiplicative margin in SphereFace is exact only when the angle between the feature and the target classifier is sufficiently small. When this angle is large, the multiplicative margin becomes approximate and the original intuition no longer holds. Motivated by this, we propose two novel variants that can exactly implement the intuition of multiplicative margin for all possible angles. Along with the new multiplicative margins, we also propose a novel implementation strategy which we call *characteristic gradient detachment* (CGD) that helps to stabilize training and improve generalization. We term our improved approach *SphereFace-R*.

Another significant difference between SphereFace and other hyperspherical FR methods [3], [4], [5], [9] is whether feature normalization (FN) is performed. Based on the empirical observation in [2], [8], we notice that feature magnitude still contains some information such as image quality. However, whether the information encoded in feature magnitude is useful for FR remains an open question. To address this, we consider three schemes here: no feature normalization (NFN), hard feature normalization (HFN) and soft feature normalization (SFN). HFN is identical to the popular feature normalization used in [3], [4], [5], [9]. In contrast, SFN formulates the feature normalization objective into a regularization term and optimizes it jointly with the neural network. Unlike HFN, SFN will take feature magnitude into account when training the neural network. This shares a similar spirit with [10]. While both FN-free learning and HFN can be viewed as limiting cases of SFN, SFN effectively unifies both approaches and serves as an interpolation between them. We conduct a systematic study to evaluate the effectiveness of all three FN strategies.

Our contributions can be summarized as follows:

- We present a unified framework to understand large angular margin in hyperspherical FR. This framework effectively explains how and why angular margin can be incorporated in SphereFace and further summarizes a general principle for loss functions to introduce large angular margin. Moreover, most of the current hyperspherical FR methods can be viewed as special instantiations of this framework.
- Under the unified framework, we substantially extend our previous work on SphereFace [2] by addressing training instability and improving empirical performance. Compared to the original SphereFace, SphereFace-R uses a more intuitive way to incorporate the multiplicative margin and yields more stable training, more clear geometric interpretation and superior generalization.
- We propose CGD, a generic implementation method for hyperspherical FR methods to improve the training stability and generalizability.
- To evaluate the usefulness of feature magnitude, we comprehensively study SphereFace-R under three different FN schemes: NFN, HFN and SFN.
- Our paper comes with an easy-to-use codebase to facilitate future research.[1] It serves as a platform to evaluate hyperspherical FR methods fairly.

2 Related Work

Deep face recognition. Deep face recognition has been an active research area in the past decades. [6], [7], [11] address open-set FR using a convolutional neural network (CNN) supervised by softmax-based loss, which essentially views open-set FR as a multi-class classification problem. [12] combines contrastive loss and softmax loss to jointly supervise the CNN training, greatly boosting performance. [13] uses triplet loss and feature normalization to learn a unified face embedding. After training on nearly 200 million face images, they achieve state-of-the-art performance. Inspired by linear discriminant analysis, [14] propose center loss for CNNs and obtain promising performance. Prior to hyperspherical FR, well-performing deep FR methods [13], [15], [16] were mostly built on either contrastive loss or triplet loss, validating the importance of solving open-set FR as a metric learning problem [2]. The development of deep FR is also closely related to deep metric learning [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28].

Hyperspherical face recognition. As a major line of research in deep FR, hyperspherical FR [2], [3], [4], [5], [9], [10], [26], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45], [46] has become increasingly popular in recent years due to its effectiveness. [1] proposes the initial framework of learning deep features with large angular margin. Built upon this framework, SphereFace [2] normalizes the classifier weights and explicitly models the decision boundary on the hypersphere. Feature normalization [9], [29], [30] has also been introduced to facilitate the learning of large angular margin face embeddings. To improve training stability, CosFace [3], [4] and ArcFace [5] propose to use additive angular margin to replace the original multiplicative margin in SphereFace and obtain impressive performance. [36] and [37] consider adaptive schemes to set the radius of the hypersphere and the margin parameter, respectively. [26], [40] study hyperspherical FR by taking the easy-hard sample balance into consideration.

Hyperspherical learning. Beyond face recognition, the idea of learning a representation on the hypersphere is also shown generally useful in a diverse set of applications, such as few-shot recognition [47], [48], [49], [50], [51], deep metric learning [25], [52], self-supervised learning [53], [54], [55], [56], generative models [57], [58], geometric learning [59], [60], [61], person re-identification [26], [62], [63], [64], speech processing [65], [66], [67], [68] and text processing [69]. It has been widely observed that constraining the embedding space on a hypersphere is beneficial to generalizability.

In contrast to hyperspherical FR methods that use an additive margin, SphereFace-R is built upon our previous work [1], [2] and adopts a multiplicative margin approach.

1. See **Project OpenSphere**: https://opensphere.world/.

TABLE 1
Instantiations of the Unified Hyperspherical Face Recognition Framework. Gray region denotes our contribution.

Method	Feature Magnitude	Non-target Function $\eta(\theta)$	Target Function $\psi(\theta)$	$\Delta(\theta)$ (shown in Fig. 1)
NormFace [9]	HFN	$\cos(\theta)$	$\cos(\theta)$	0
CosFace [3], [4]	HFN	$\cos(\theta)$	$\cos(\theta) - m$	m
ArcFace [5]	HFN	$\cos(\theta)$	$\cos(\theta + m)$	$\cos(\theta) - \cos(\theta + m)$
SphereFace	NFN [2]/HFN/SFN	$\cos(\theta)$	$(-1)^k \cos(m\theta) - 2k$, $\theta \in [\frac{k\pi}{m}, \frac{(k+1)\pi}{m}]$, $k \in \mathbb{N}$	$\cos(\theta) - (-1)^k \cos(m\theta) + 2k$
SphereFace-R v1	NFN/HFN/SFN	$\cos(\theta)$	$\cos(\min\{m, \frac{\pi}{\theta}\} \cdot \theta)$	$\cos(\theta) - \cos(\min\{m, \frac{\pi}{\theta}\} \cdot \theta)$
SphereFace-R v2	NFN/HFN/SFN	$\cos(\frac{\theta}{m})$	$\cos(\theta)$	$\cos(\frac{\theta}{m}) - \cos(\theta)$

3 A Unified Large-Margin Learning Framework for Hyperspherical Face Recognition

To gain deeper insights towards large angular margin, we present a unified framework for hyperspherical FR. To start with, we consider the standard softmax cross-entropy loss:

$$\mathcal{L}_s = -\log\left(\frac{\exp(\boldsymbol{W}_y^\top \boldsymbol{x} + b_y)}{\sum_{i=1}^{K} \exp(\boldsymbol{W}_i^\top \boldsymbol{x} + b_i)}\right) \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^d$ denotes the deep feature (the input of the classifier layer), y is its ground truth label, K is the total number of classes, $\boldsymbol{W}_i \in \mathbb{R}^d$ is the weights of the i-th classifier and b_i is the bias for the i-th class. Note that here we consider the case of a single input sample for simplicity; we only need to average the loss objectives if we consider a mini-batch of input samples. Since the class-dependent bias term is not informative in open-set evaluation, we follow the common practice to remove it [2]. Then we normalize the classifier weights to one (*i.e.*, $\|\boldsymbol{W}_i\| = 1, \forall i$) and rewrite the objective function as follows:

$$\mathcal{L}_s = -\log\left(\frac{\exp\left(\|\boldsymbol{x}\| \cos(\theta_y)\right)}{\sum_{i=1}^{K} \exp(\|\boldsymbol{x}\| \cos(\theta_i))}\right) \tag{2}$$

where θ_i denotes the angle between deep feature $\boldsymbol{x}$ and the i-th classifier $\boldsymbol{W}_i$. By considering a generic angular activation rather than the cosine function, we have the following generalized objective function:

$$\mathcal{L}_g = -\log\left(\frac{\exp\left(\|\boldsymbol{x}\| \psi(\theta_y)\right)}{\exp\left(\|\boldsymbol{x}\| \psi(\theta_y)\right) + \sum_{i \neq y} \exp(\|\boldsymbol{x}\| \eta(\theta_i))}\right) \tag{3}$$

where $\psi(\theta_y)$ is the angular activation function for the target class (*i.e.*, ground truth label) and $\eta(\theta_i), i \neq y$ denotes the angular activation function for the i-th non-target class (the labels excluding the ground truth one). Similar to the cosine function, both $\psi(\theta)$ and $\eta(\theta)$ are generally required to be monotonically decreasing for $\theta \in [0, \pi]$. After looking into different hyperspherical FR methods, we summarize a simple yet generic principle for any softmax loss in order to learn embeddings with large angular margin.

> To achieve large angular margin, the generic principle is to make $\psi(\theta)$ always smaller than $\eta(\theta)$ in $(0, \pi]$, namely
>
> $$\Delta(\theta) = \eta(\theta) - \psi(\theta) > 0 \tag{4}$$
>
> where we define $\Delta(\theta)$ as the characteristic function for large angular margin. $\Delta(\theta)$ determines most of the properties about the angular margin, such as the size of the margin, its learning stability, etc.

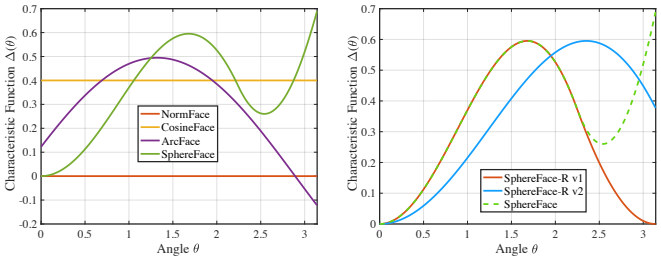
As long as we guarantee that $\Delta(\theta)$ is larger than zero, then the objective function in Eq. (3) will define a task that can inherently introduce large angular margin. To see how $\Delta(\theta)$ interacts with the loss function, we can rewrite Eq. (3) in the following mathematically equivalent form:

$$\begin{aligned}\mathcal{L}_g &= \log\left(1 + \sum_{i \neq y} \exp\left(\|\boldsymbol{x}\| \left(\eta(\theta_i) - \psi(\theta_y)\right)\right)\right) \\ &= \log\left(1 + \sum_{i \neq y} \exp\left(\|\boldsymbol{x}\| \left(\eta(\theta_i) - \eta(\theta_y) + \Delta(\theta_y)\right)\right)\right)\end{aligned} \tag{5}$$

which essentially aims to minimize $\eta(\theta_i) - \eta(\theta_y) + \Delta(\theta_y)$. The term $\eta(\theta_i) - \eta(\theta_y)$ represents the difference of classification confidence, and the characteristic function $\Delta(\theta_y)$ controls the angular margin. When $\Delta(\theta_y) = 0$ and $\eta(\cdot)$ is the cosine function, Eq. (5) reduces to the standard softmax loss with weight normalization. $\Delta(\theta_y) = 0$ indicates that no angular margin has been introduced. When $\Delta(\theta_y) > 0$, this leads to large angular margin because it makes the classification more stringent (*i.e.*, the neural network will learn to make θ_y smaller in order to reach the same loss value as the case of $\Delta(\theta_y) = 0$). It is also worth mentioning that when $\Delta(\theta_y) < 0$, Eq. (5) defines an easier task than the standard classification problem and is potentially useful for robust learning against noisy images or labels. Our paper focuses on the case of $\Delta(\theta_y) > 0$.

We note that there exist scenarios where large angular margin can still be achieved even if $\Delta(\theta_y)$ is smaller than zero in some range of $\theta_y \in [0, \theta]$. For example, $\Delta(\theta_y)$ for ArcFace can be smaller than zero when θ_y is close to π. ArcFace can still introduce angular margin because the case where θ_y is close to π hardly happens with real data distribution, as verified by [5]. Nonetheless, the characteristic function for ArcFace still approximately satisfies our principle, since it is larger than zero with most $\theta_y \in [0, \theta]$. Therefore, as long as the characteristic function $\Delta(\theta_y)$ is larger than zero for the angles where θ_y is densely distributed in practice (*i.e.*, $\mathbb{E}_{\theta_y} \Delta(\theta_y) > 0$), it will typically suffice to produce effective angular margin. Our principle in fact serves as a sufficient condition to introduce angular margin. It is generally better to use our principle as the guideline for designing new angular margin losses, because the empirical distribution of the target angle θ_y could vary under difference circumstances (*e.g.*, network architectures, datasets, optimizers).

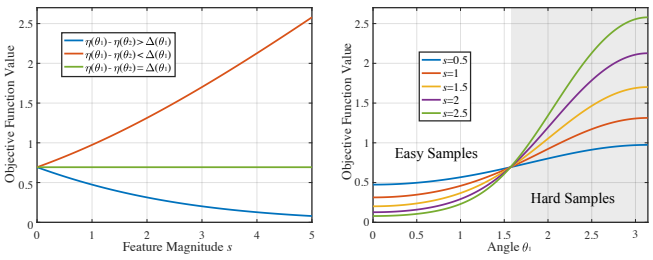
We now discuss in depth why $\Delta(\theta_y) > 0$ is able to introduce large angular margin. For ease of illustration, we consider the binary case where the first class is the target class. In this case, we only need to discuss $\eta(\theta_2) - \eta(\theta_1) + \Delta(\theta_1)$. If $\Delta(\theta_1) = 0$, the decision boundary



(a) Comparison between SphereFace and existing methods

(b) Comparison between SphereFace and SphereFace-R v1/v2

Fig. 1. (a) $\Delta(\theta)$ for current representative hyperspherical FR methods. (b) $\Delta(\theta)$ of SphereFace and SphereFace-R. We set m to 0.4, 0.5, 1.4, 1.4 and 1.4 for CosFace, ArcFace, SphereFace, SphereFace-R v1 and SphereFace-R v2, respectively.



(a) Loss value vs. feature magnitude under different angular scenarios

(b) Loss value vs. angle under different feature magnitude

Fig. 2. (a) How the loss value changes as feature magnitude s increases under different angular scenarios in Eq. (7). For this figure, we consider the binary case where $\theta_1 = \pi/3$ ($y = 1$) and $\theta_2 = \pi/2$. Both $\eta(\cdot)$ and $\psi(\cdot)$ are cosine function. (b) How the loss curve of Eq. (6) varies under different feature magnitude s. For this figure, we consider the binary case where $y = 1$ and $\theta_2 = \pi/2$.

for the first class is $\eta(\theta_2) - \eta(\theta_1) = 0$ which is equivalent to $\theta_1 = \theta_2$. When $\theta_1 < \theta_2$, the sample $\boldsymbol{x}$ will be classified to the first class. If $\Delta(\theta_1) > 0$ and $\eta(\cdot)$ is monotonically decreasing, then the decision boundary for the first class becomes $\eta(\theta_2) - \eta(\theta_1) + \Delta(\theta_1) = 0$ which is equivalent to $\theta_1 + m(\theta_1) = \theta_2$ where $m(\cdot)$ denotes some positive function (the specific form of $m(\cdot)$ is determined by $\eta(\cdot)$ and $\Delta(\cdot)$, but it stays positive as long as $\Delta(\cdot)$ is always positive). Therefore, now we need to make $\theta_1 + m(\theta_1) < \theta_2$ in order to classify $\boldsymbol{x}$ to the first class, and the decision boundary for the first class becomes more stringent than the previous case. The neural network has to learn smaller θ_1 in order to correctly classify $\boldsymbol{x}$ and smaller θ_1 implies a more compact representation for the first class. The same reasoning also applies to the case where $\boldsymbol{x}$ belongs to the second class (*i.e.*, the second class is the target class). As a result, if we can successfully train a neural network to correctly classify training samples with these more stringent classification criteria, $\Delta(\theta_1) > 0$ can effectively produce large angular margin for the learned deep features.

Importantly, current popular hyperspherical FR methods can be viewed as special cases under this unified framework, as shown in Table 1 (first four rows). To intuitively understand different variants of angular margin, we also compare their characteristic functions $\Delta(\theta)$ in Fig. 1(a). One can observe that different hyperspherical FR methods yield different large-margin characteristic functions. Each characteristic function determines how a hyperspherical FR method performs and therefore it is of great significance to design a suitable characteristic function. Specifically, the characteristic function $\Delta(\theta)$ clearly reveals the induced angular margin for samples with different recognition hardness (larger θ_y typically implies a harder sample). Instead of a static characteristic function, designing a dynamic characteristic function could be beneficial [37], [40]. It is also possible to learn the characteristic function in a data-driven and automatic fashion, as explored in [70], [71].

Besides the characteristic function $\Delta(\cdot)$, the feature magnitude $\|\boldsymbol{x}\|$ in Eq. (5) also plays a non-negligible role in learning large angular margin. The original SphereFace approach preserves the feature magnitude in training, since the feature magnitude does not affect the angular decision boundary. [3], [4], [5], [9], [29] show that normalizing the feature magnitude to a constant s (*e.g.*, making $\boldsymbol{x} \leftarrow s\frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}$ in Eq. (5)) can stabilize training and also improve hyperspherical discriminativeness. By normalizing the feature magnitude to a prescribed positive constant s, Eq. (5) becomes

$$\mathcal{L}_s = \log\left(1 + \sum_{i \neq y} \exp\left(s \cdot (\eta(\theta_i) - \eta(\theta_y) + \Delta(\theta_y))\right)\right) \quad (6)$$

where s is a universal value instead of the original instance-dependent $\|\boldsymbol{x}\|$. There are two advantages of feature normalization. First, it can effectively avoid potential bad local minima. Second, it can help the loss function to better balance easy and hard training samples.

For the first aspect, we consider Eq. (6) in a simple binary classification scenario (class 1 is the ground truth label for the deep feature $\boldsymbol{x}$, *i.e.*, $y = 1$). The loss value can easily go to zero once the deep feature $\boldsymbol{x}$ lies in the correct decision region, as demonstrated in the following equation:

$$\begin{aligned} &\lim_{s \to +\infty} \log\left(1 + \exp\left(s \cdot (\eta(\theta_2) - \eta(\theta_1) + \Delta(\theta_1))\right)\right) \\ &= \begin{cases} 0 & \text{if} \quad \eta(\theta_1) - \eta(\theta_2) > \Delta(\theta_1) \\ -\log\frac{1}{2} & \text{if} \quad \eta(\theta_1) - \eta(\theta_2) = \Delta(\theta_1) \\ +\infty & \text{if} \quad \eta(\theta_1) - \eta(\theta_2) < \Delta(\theta_1) \end{cases} \end{aligned} \quad (7)$$

where $\eta(\theta_1) - \eta(\theta_2) > \Delta(\theta_1)$ means that $\boldsymbol{x}$ can be correctly classified, $\eta(\theta_1) - \eta(\theta_2) = \Delta(\theta_1)$ means that $\boldsymbol{x}$ exactly lies on the decision boundary and $\eta(\theta_1) - \eta(\theta_2) < \Delta(\theta_1)$ means that $\boldsymbol{x}$ can not be correctly classified. The results imply that when $\boldsymbol{x}$ can be correctly classified, a trivial solution to reduce loss to zero is to simply increase s. However, increasing s does not help the neural network learn angularly discriminative face embeddings and results in bad local minima. Because $\|\boldsymbol{x}\|$ can be viewed as an instance-dependent learnable s, the neural network without feature normalization is likely to simply increase $\|\boldsymbol{x}\|$ after θ_1 passes the decision boundary. Therefore, using a constant s can prevent this trivial way of reducing loss value and eliminate these bad local minima. We also plot how the loss value changes as s increases in Fig. 2(a). The same argument can easily generalize to the multi-class scenario, as shown in

$$\begin{aligned} &\lim_{s \to +\infty} \log\left(1 + \sum_{i \neq y} \exp\left(s(\eta(\theta_i) - \eta(\theta_y) + \Delta(\theta_y))\right)\right) \\ &= \begin{cases} 0 & \text{if} \quad \forall i \neq y,\ \eta(\theta_y) - \eta(\theta_i) > \Delta(\theta_y) \\ +\infty & \text{if} \quad \exists i \neq y,\ \eta(\theta_y) - \eta(\theta_i) < \Delta(\theta_y) \end{cases} \end{aligned} \quad (8)$$

where s has a large influence on the loss value. For the multi-class scenario, the neural network can trivially in-

crease s to minimize the loss once $\eta(\theta_y) - \eta(\theta_i) > \Delta(\theta_y)$ for all $i \neq y$. Interestingly, this also explains why the standard softmax loss cannot learn deep features with large angular margin. Empirically the standard softmax loss tends to increase s instead of minimizing the target angle once the deep feature $\boldsymbol{x}$ falls into the correct decision boundary, leading to separable features rather than large-margin features. Large-margin losses take advantage of this phenomenon and make the decision boundary asymmetric for different classes (*i.e.*, $\eta(\theta) \neq \psi(\theta)$). Then the classification of $\boldsymbol{x}$ (*i.e.*, forcing $\psi(\theta_y) > \eta(\theta_i), \forall i \neq y$) naturally becomes equivalent to learning large-margin deep features.

For the second aspect, we use an example to demonstrate how the feature magnitude s can balance the easy and hard samples. We compare the loss function under different s in Fig. 2(b). By adjusting s, the loss function in Eq. (6) has different sensitivity for samples with different target angle θ_y. Intuitively, samples with large target angle are considered to be hard, while samples with small target angle are viewed as easy. Therefore, feature magnitude s can also balance the loss value for easy and hard samples, which serves a role similar to hard sample mining [21] in deep metric learning. From Fig. 2(b), one can observe that larger s puts more focus on the hard samples, since the loss ratio between hard and easy samples increases. Finding a good s essentially can be viewed as searching for a suitable balance between easy and hard samples in hyperspherical FR methods, *e.g.*, [3], [4], [5], [9].

To summarize, Eq. (6) essentially throws away the information encoded in the feature magnitude $\boldsymbol{x}$. Despite the two major advantages that ease the training of hyperspherical FR methods, it remains an open problem whether it is beneficial to combine feature magnitude to training. Feature magnitude is closely related to image quality and semantic ambiguity [8], [72], and such information intuitively seems useful to distinguish different faces. However, training hyperspherical FR methods without feature normalization generally yields inferior training stability and generalization performance in practice. In order to explore whether feature magnitude is indeed helpful or not, we consider to constrain the feature magnitude via a soft regularization in Section 4.2. This serves as an interpolation between no feature normalization and hard feature normalization, and can take the feature magnitude into account during training.

4 SphereFace-R: Better and More Stable

In this section, we elaborate the design of SphereFace-R and introduce two novel variants that perform well in practice. Sharing the same geometric interpretation as SphereFace, SphereFace-R yields improved training stability and superior open-set generalizability. We start by revisiting the design of the original SphereFace and then propose alternative ways to implement the multiplicative margin in Section 4.1. In Section 4.2, we discuss different feature normalization schemes. Finally, we list a few important open problems for hyperspherical FR in Section 4.3.

4.1 Rethinking Multiplicative Angular Margin

As has been shown in Table 1, all the previous hyperspherical FR methods focus on designing a good target angular

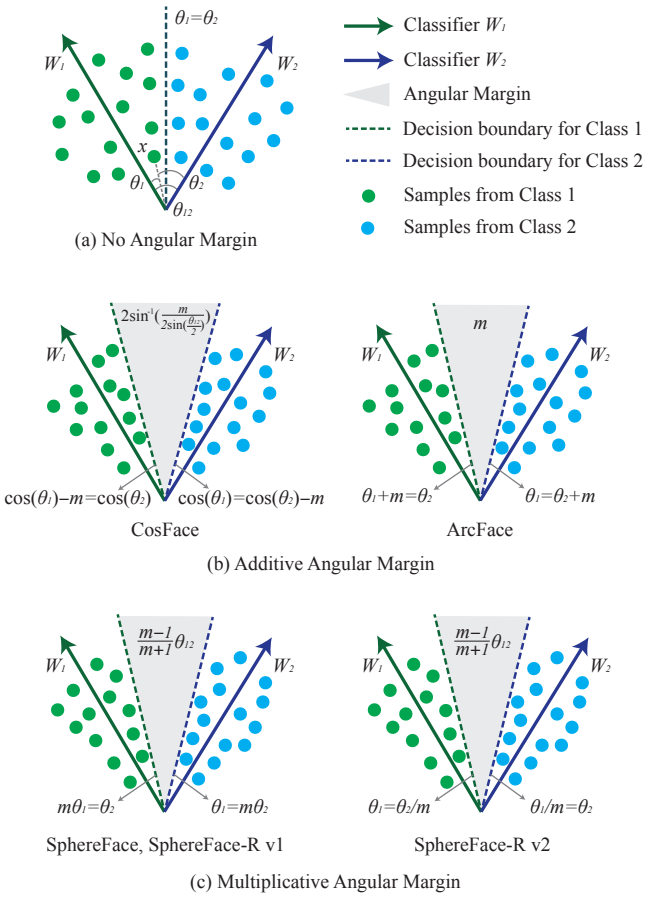


Fig. 3. An intuitive comparison among no angular margin (*e.g.*, [9], [30]), additive angular margin (CosFace [3], [4] and ArcFace [5]) and multiplicative angular margin (SphereFace, SphereFace-R v1 and SphereFace-R v2).

activation function. SphereFace adopts the same paradigm by constructing the following target angular function $\psi(\theta)$:

$$\psi(\theta) = (-1)^k \cos(m\theta) - 2k, \quad \theta \in \left[\frac{k\pi}{m}, \frac{(k+1)\pi}{m}\right] \tag{9}$$

where $k \in [0, m-1]$ and k is an integer. After ensuring $\eta(\theta) = \cos(\theta)$ and Eq. (9), Eq. (5) becomes the objective function for the original SphereFace, as summarized in Table 1. The original SphereFace requires m to be an integer, which is in fact unnecessary. m can be any positive value larger than 1. In order to improve training stability, our original SphereFace minimizes its objective function (with $m = 4$) jointly with a standard softmax loss, which approximately yields an effective m as 1.4. Therefore, the target angular function in the original SphereFace can be simplified to

$$\psi(\theta) = \begin{cases} \cos(m\theta) & \text{if} \quad 0 \leq \theta \leq \frac{\pi}{m} \\ -\cos(m\theta) - 2 & \text{if} \quad \frac{\pi}{m} < \theta \leq \pi \end{cases} \tag{10}$$

where we usually use $m \in [1, 2]$. One can easily verify that the characteristic function $\Delta(\theta)$ is always larger than zero with $\theta \in (0, \pi]$, so it satisfies the general principle to introduce large angular margin. Fig. 3 intuitively compares no angular margin, additive angular margin and multiplicative angular margin. The intuition of multiplicative margin can be understood from a simple binary classification example (with two classifiers $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$). We consider Eq. (3) with

$\eta(\theta) = \cos(\theta)$ and $\psi(\theta) = \cos(\theta)$. For a sample $\boldsymbol{x}$, we need to require $\cos(\theta_1) > \cos(\theta_2)$ to correctly classify $\boldsymbol{x}$. But what if we instead require $\cos(m\theta_1) > \cos(\theta_2)$ where $m > 1$ in order to correctly classify $\boldsymbol{x}$? It is essentially making the decision more stringent than previous, because we require a lower bound[2] of $\cos(\theta_1)$ to be larger than $\cos(\theta_2)$. The decision boundary for class 1 is $\cos(m\theta_1) = \cos(\theta_2)$. Similarly, if we require $\cos(m\theta_2) > \cos(\theta_1)$ to correctly classify samples from class 2, the decision boundary for class 2 is $\cos(m\theta_2) = \cos(\theta_1)$. Suppose all training samples are correctly classified, such asymmetric decision boundaries will naturally produce an angular margin of size $\frac{m-1}{m+1}\theta_{12}$ where θ_{12} denotes the angle between $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$. From angular perspective, correctly classifying $\boldsymbol{x}$ from identity 1 requires $\theta_1 < \frac{\theta_2}{m}$, while correctly classifying $\boldsymbol{x}$ from identity 2 requires $\theta_2 < \frac{\theta_1}{m}$. If $m > 1$, both decision criteria are more difficult to achieve than the vanilla case without any angular margin (*i.e.*, $\theta_1 < \theta_2$ and $\theta_2 < \theta_1$).

We can observe that Eq. (10) can exactly match the intuition of multiplicative margin only when $\theta \in [0, \frac{\pi}{m}]$. When $\theta \in (\frac{\pi}{m}, \pi]$, the same argument however will no longer hold. Although such a heuristic design can still empirically achieve large angular margin and work reasonably well, it may inevitably be less interpretable and also contribute to the training instability. The key to multiplicative angular margin is to guarantee that the equation $\psi(\theta) = \eta(m\theta)$ ($m > 1$) always holds for $\theta \in [0, \pi]$. In order to better implement the intuition of multiplicative margin, we propose two different approaches, *i.e.*, designing either a new target angular function $\psi(\theta)$ or a new non-target function $\eta(\theta)$. In Section 4.1.1, we first follow the original idea of SphereFace to re-design a target angular function $\psi(\theta)$ which can better reflect the intuition of multiplicative margin. In Section 4.1.2, we take a different approach by designing a new $\eta(\theta)$ which has a much simpler form yet can exactly match the intuition of multiplicative margin for $\theta \in [0, \pi]$. Section 4.1.3 proposes a useful implementation method to further stabilize training. Section 4.1.4 gives implications and discussions.

4.1.1 SphereFace-R v1: On Designing $\psi(\theta)$

Following the conventional way to design an angular margin loss [2], [3], [4], [5], we first focus on constructing a target angular function $\psi(\theta)$ based on the intuition of multiplicative margin. For $\theta \in [0, \frac{\pi}{m}]$, we can simply use $\psi(\theta) = \cos(m\theta)$ which is a monotonically decreasing function in $[0, \frac{\pi}{m}]$ and exactly implements the multiplicative angular margin. When $\theta > \frac{\pi}{m}$, SphereFace constructs a surrogate monotonically decreasing function to replace $\cos(m\theta)$, as specified in Eq. (10). However, this design of $\psi(\theta)$ in $[\frac{\pi}{m}, \pi]$ does not follow the original intuition of multiplicative margin and may be sub-optimal. In order to better implement multiplicative margin in the entire domain of $[0, \pi]$, we propose the following target angular function:

$$\psi(\theta) = \cos\left(\min\left\{m, \frac{\pi}{\theta}\right\} \cdot \theta\right) \tag{11}$$

where m is usually a prescribed positive constant. Eq. (11) remains a monotonic function in $[0, \pi]$ and can be viewed as incorporating large angular margin with a dynamic multiplicative parameter $\min\{m, \frac{\pi}{\theta}\}$. For $\theta \in [0, \frac{\pi}{m}]$, Eq. (11) is exactly the same as SphereFace and perfectly implements the multiplicative margin. For $\theta \in [\frac{\pi}{m}, \pi]$, we consider a new multiplicative margin parameter m' and the target angular function becomes $\psi(\theta) = \cos(m'\theta)$. In order to *(i)* make m' as large as possible and *(ii)* make $\psi(\theta)$ a monotonic decreasing function where $m'\theta$ does not exceed π, we propose an adaptive decreasing strategy for m': $m' = \frac{\pi}{\theta}$. Combining pieces, we end up with a dynamic multiplicative margin parameter $m' = \min\{m, \frac{\pi}{\theta}\}$. The non-target angular function is the same as SphereFace, *i.e.*, $\eta(\theta) = \cos(\theta)$. Therefore, the multiplicative margin is implemented through $\psi(\theta) = \eta(m'\theta)$. The curve of the corresponding characteristic function $\Delta(\cdot)$ is given in Fig. 1(b). More interestingly, we can observe that SphereFace-R v1 incorporates less angular margin to samples that are too easy or too hard (*i.e.*, the target angle is around 0 or π) and combines the largest angular margin to samples with medium hardness. We also compare SphereFace-R v1 with the other hyperspherical FR methods in Table 1.

Despite the well implemented multiplicative margin in Eq. (11), there is still a constraint on the effective multiplicative margin parameter m', *i.e.*, $m' \leq \frac{\pi}{\theta}$. Moreover, we have no consistent m' in Eq. (11) for samples with different target angle. It indicates that for an arbitrary sample whose target angle is within $[0, \pi]$, SphereFace-R v1 can not guarantee the same m'. To address this limitation, we propose to implement the multiplicative margin from the perspective of the non-target angular function rather than the target angular function, leading to SphereFace-R v2.

4.1.2 SphereFace-R v2: On Designing $\eta(\theta)$

We consider how to design the non-target angular function $\eta(\theta)$ based on the intuition of multiplicative margin. To achieve $\psi(\theta) = \eta(m\theta)$ without changing the target angular function $\psi(\theta)$, we can naturally arrive at the following desired non-target angular function $\eta(\theta)$:

$$\eta(\theta) = \psi\left(\frac{\theta}{m}\right) = \cos\left(\frac{\theta}{m}\right) \tag{12}$$

where m is a prescribed positive constant and $\psi(\theta) = \cos(\theta)$. Compared to Eq. (11) in SphereFace-R v1, Eq. (12) is much simpler and more importantly, satisfies the property of $\psi(\theta) = \eta(m\theta)$ for $\theta \in [0, \pi]$ with a static m. While being extremely simple and conceptually appealing, SphereFace-R v2 can also exactly incorporate a static multiplicative angular margin. In contrast, SphereFace-R v1 is unable to induce a static multiplicative margin with $m > 1$ and can only incorporate a dynamic multiplicative margin where the effective margin parameter has to be close to 1 if θ is near π. More importantly, unlike SphereFace-R v1, there is no constraint for the size of the induced angular margin in SphereFace-R v2 and we can use any desirable $m \geq 1$.

From the corresponding characteristic function given in Fig. 1(b), we can see that SphereFace-R v2 incorporates the smallest angular margin to easiest samples and the largest angular margin to samples with medium hardness. Unlike SphereFace-R v1 that introduces very small angular margin to hard samples, SphereFace-R v2 combines much larger angular margin to these samples. Therefore, SphereFace-R v1 and SphereFace-R v2 put different efforts on optimizing hard samples and may yield different generalizability.

2. The inequality $\cos(\theta_1) > \cos(m\theta_1)$ holds if $\theta_1 \in [0, \frac{\pi}{m}], m > 1$.

To the best of our knowledge, SphereFace-R v2 is the very first method that introduces large angular margin through non-target angular functions. SphereFace-R v2 easily addresses the difficult problem of incorporating a static multiplicative margin by simply switching the design focus from target function to non-target function. We believe that this method provides an important and novel perspective on designing large angular margin losses.

4.1.3 Characteristic Gradient Detachment

In order to further stabilize training and improve performance, we introduce a simple and generic method – characteristic gradient detachment for implementing our multiplicative margin. In general, the shape of the characteristic function $\Delta(\theta)$ determines the training stability and the convergence property. Empirically, we find that a characteristic function with simpler backward gradient computation typically leads to better training stability. For example, CosFace [3], [4] yields strong empirical training stability and its characteristic function is simply a positive constant with backward gradient as 0. Inspired by such an observation, we aim to simplify the backward gradient computation for the characteristic function. To gain more intuitions, we first use Taylor expansion to decompose the characteristic function in the target angular function at an arbitrary angle $\theta_0 \in (0, \pi)$ with a small angle deviation $\delta\theta$:

$$\begin{aligned}\psi(\theta_0+\delta\theta) &= \eta(\theta_0+\delta\theta) - \Delta(\theta_0+\delta\theta) \\ &= \eta(\theta_0+\delta\theta) - \big(\Delta(\theta_0) + \frac{\Delta'(\theta_0)}{1!}\delta\theta + \\ &\quad \cdots + \frac{\Delta^{(n)}(\theta_0)}{n!}(\delta\theta)^n + R_n(\delta\theta)\big)\end{aligned} \tag{13}$$

where $\Delta^{(n)}(\theta)$ denotes the n-th order derivative of $\Delta(\theta)$ and $R_n(\delta\theta)$ denotes the higher order infinitesimal of $(\delta\theta)^n$. When the characteristic function is more complex, then its Taylor expansion needs to have more terms to accurately represent it. This leads to more complex backward gradient computation. Motivated by the observation that simpler gradient computation often leads to better training stability, we propose to make an approximation to the characteristic function by removing some higher order terms in its Taylor expansion. Generally, we can remove any higher order Taylor expansion terms and it yields different backward gradients. Particularly, we draw inspirations from the constant characteristic function adopted in CosFace, and use the zero-order approximation for $\Delta(\theta_0+\delta\theta)$ in Eq. (13):

$$\psi(\theta_0+\delta\theta) \approx \eta(\theta_0+\delta\theta) - \Delta(\theta_0) \tag{14}$$

which is much simpler and robust to compute and gives the following approximate gradient for $\psi(\theta)$ at θ_0:

$$\psi'(\theta_0) = \lim_{\delta\theta\to 0}\frac{\psi(\theta_0+\delta\theta)-\psi(\theta_0)}{\delta\theta} \approx \eta'(\theta_0) \tag{15}$$

which naturally leads to the proposed CGD where we can simply apply gradient detachment to the characteristic function $\Delta(\theta)$. Specifically, we stop the gradient of the characteristic function with a detachment operator:

$$\psi(\theta) = \eta(\theta) - \text{Detach}(\Delta(\theta)) \tag{16}$$

where $\text{Detach}(\cdot)$ denotes the detachment operator that allows forward computation but stops the backward gradient propagation. This essentially means that we only need to compute the characteristic function in the forward pass and completely ignore it in the backward propagation. In order to avoid computing the gradient of the characteristic function, we substitute Eq. (16) into the first line of Eq. (5) and finally obtain the following loss function:

$$\mathcal{L}_{\text{v1}} = \log\Big(1+\sum_{i\neq y}\exp\big(\|\boldsymbol{x}\|\,(\eta(\theta_i)-\eta(\theta_y)+\text{Detach}(\Delta(\theta_y)))\big)\Big)$$

which can be generally used for the cases where the target angular function is modified, such as SphereFace and SphereFace-R v1. In the backward pass, CGD approximates the characteristic function of the multiplicative margin with a piece-wise function consisting of many constant functions, as illustrated in Fig. 4. Equivalently, CGD can also be viewed as a step function approximation to the characteristic function in the backward pass. We note that the approximation in CGD only exists in the backward direction and the forward computation is always identical to the CGD-free scenario. From a different perspective, CGD can be understood as interpreting the multiplicative margin with a dynamic CosFace-style additive margin (*i.e.*, the effective margin parameter for the additive margin is dynamically dependent on the input target angle in CGD rather than being static in CosFace). The discussion above applies to both SphereFace and SphereFace-R v1, since they are modifying the target angular function. As a concrete example, applying CGD to SphereFace-R v1 yields the target angular function: $\psi(\theta) = \cos(\theta) - \text{Detach}(\cos(\theta) - \cos(\min\{m, \frac{\pi}{\theta}\}\cdot\theta))$ whose gradient is identical to CosFace.

For SphereFace-R v2 that modifies the non-target angular function, the derivation is similar except that we focus on approximating the gradient of the non-target function $\eta(\theta)$ instead of the target function $\psi(\theta)$. Therefore, we can

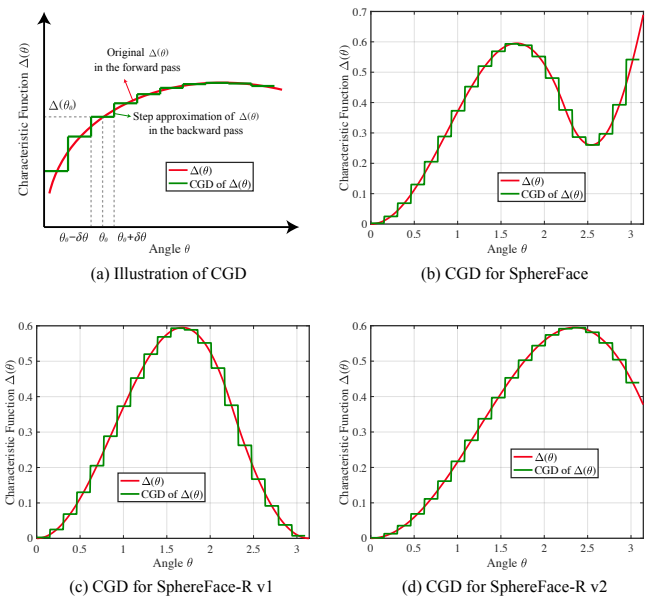


Fig. 4. An illustration of the backward activation of CGD for SphereFace ($m = 1.4$), SphereFace-R v1 ($m = 1.4$) and SphereFace-R v2 ($m = 1.4$). The green curves demonstrate the effect of CGD in the backward propagation, and the forward computation still follows the red curves (*i.e.*, $\Delta(\theta)$) without any approximation.

similarly apply gradient detachment to the characteristic function in the non-target angular function:

$$\eta(\theta) = \psi(\theta) + \text{Detach}(\Delta(\theta)). \tag{17}$$

After putting Eq. (17) into the first line of Eq. (5), we end up with the following general loss function for the cases that modify the non-target function:

$$\mathcal{L}_{\text{v2}} = \log\Big(1+\sum_{i\neq y}\exp\big(\|\boldsymbol{x}\|\,(\psi(\theta_i)-\psi(\theta_y)+\text{Detach}(\Delta(\theta_i)))\big)\Big)$$

from which we can see that the key is to detach the gradients of the characteristic function. Therefore, applying CGD to SphereFace-R v2 yields the non-target function: $\eta(\theta) = \cos(\theta) + \text{Detach}(\cos(\frac{\theta}{m}) - \cos(\theta))$. The essence of CGD is to avoid computing the gradient of the characteristic function in the backward pass. As a simple generalization, we can consider higher-order Taylor approximation to the characteristic function instead of the zero-order approximation. Since CGD already yields satisfactory training stability and empirical performance, we will stick to it for simplicity.

In fact, CGD serves as a generally useful tool for implementing new types of angular margin and is not limited to SphereFace and SphereFace-R. For the backward propagation, CGD can approximate the characteristic function induced by any angular margin with a dynamic additive margin, and effectively stabilize the training.

4.1.4 Implications and Discussions

Comparison between additive and multiplicative margin. While Table 1 provides a detailed comparison between additive and multiplicative margin, the fundamental difference between them is on a conceptual level. Additive margin is introduced by adding or subtracting a parameter to the target function so that the characteristic function $\Delta(\theta)$ can be larger than zero in most cases. Specifically, this parameter can be either inside [5] or outside [3], [4] the cosine function. In contrast, multiplicative margin is achieved by multiplying a parameter to the target or non-target function so that $\Delta(\theta)$ is larger than zero. It is also possible that a multiplicative margin loss and an additive margin loss lead to the same characteristic function, and they may be technically the same loss. Therefore, their difference is determined by the specific intuition that guides the loss design.

Generality of multiplicative margin. SphereFace-R v1 and v2 demonstrate two different strategies to incorporate multiplicative angular margin, showing the existence of many feasible designs to achieve multiplicative margin. In fact, the exact form of the loss function is not crucial and the core of multiplicative margin lies in the spirit of multiplying a factor to ensure the characteristic function to be larger than zero. Following such a spirit, there are likely many potential loss designs that can work as well as ours.

Comparison between SphereFace and SphereFace-R. It is easy to see that SphereFace employs a surrogate characteristic function to implement the multiplicative margin and does not follow the intuition of multiplicative margin for $\theta \in [\frac{\pi}{m}, \pi]$ where $m > 1$. In contrast, both SphereFace-R v1 and v2 exactly follow the intuition of multiplicative margin in the entire domain of $[0, \pi]$. SphereFace-R v1 implements a dynamic multiplicative margin (*i.e.*, the effective margin parameter varies depending on the training sample), while SphereFace-R v2 implements a static one (*i.e.*, the effective margin parameter stays the same for all training samples). Moreover, SphereFace and SphereFace-R use different effective margin parameters for samples with different hardness. Both SphereFace and SphereFace-R strictly satisfy the general principle in Eq. (4) to achieve large angular margin, validating the effectiveness of our proposed principle.

Jointly designing target and non-target functions. Because SphereFace-R v1 focuses on the target angular function and SphereFace-R v2 focuses on the non-target angular function, it is natural to consider to simultaneously design the target and non-target angular functions. For example, Eq. (11) and Eq. (12) can be easily used together and the resulting characteristic function is simply the combination of both. More interestingly, it is not necessary for both target and non-target functions to use the cosine-based design. We can simply use a linear function as the target and non-target functions, as proposed in [31]. It can effectively alleviate some design constraints caused by the periodicity of cosine function. Jointly designing the target and non-target functions can greatly enlarge the search space of the characteristic function and may lead to a better multiplicative margin loss.

Beyond additive and multiplicative margin. There are many more alternative types of angular margin other than the additive and multiplicative ones. For example, we can also use the exponential function to achieve $\Delta(\theta) > 0$. Specifically, we use $\eta(\theta) = \cos(2\theta)$ and $\psi(\theta) = \cos^m(2\theta)$, where $m > 1$ is the margin parameter and larger m gives larger angular margin. Alternatively, we can also combine additive and multiplicative margin as $\eta(\theta) = \cos(\theta), \psi(\theta) = \cos(m_1\theta + m_2) - m_3$. It remains an open problem to design a simple yet well-performing angular margin.

4.2 Feature Magnitude

SphereFace [2] originally does not use feature normalization, because the feature magnitude does not affect the angular decision boundary. [3], [4], [5], [9] show that feature normalization can ease the difficulty of minimizing angular margin losses and greatly improve the training stability. Despite being effective to stabilize training, feature normalization inevitably loses useful information about individual samples (*e.g.*, image quality). Existing hyperspherical FR methods either preserve the feature magnitude in the loss function [1], [2], [32] or normalize the feature magnitude to constant s [9], [30]. To explore whether feature magnitude can be beneficial to generalization, we systematically study SphereFace and SphereFace-R under NFN and HFN. Moreover, we consider a soft feature normalization method which effectively unifies NFN and HFN and serves as an interpolation between both.

Hard feature normalization. HFN becomes a default component in current hyperspherical FR methods [3], [4], [5], [26], [40]. By normalizing the feature $\boldsymbol{x}$ to a constant s, the objective function value will merely depend on the angles between $\boldsymbol{x}$ and the classifiers $\boldsymbol{W}_i, \forall i$. In order to perform such a hard normalization on the feature $\boldsymbol{x}$, we parameterize the original $\boldsymbol{x}$ in Eq. (5) with $s\frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}$ and arrive at Eq. (6). Since s is a prescribed constant, it is equivalent to normalizing all the features to a hypersphere with radius s.

Soft feature normalization as an interpolation. We consider the soft feature normalization that interpolates between FN-free learning and HFN. Specifically, besides the original loss, we combine an additional regularization term to constrain the feature magnitude:

$$\mathcal{L}_{\text{SFN}} = t \cdot \left\| \|\boldsymbol{x}\| - s \right\|^2 \tag{18}$$

where t is a hyperparameter that controls the regularization strength and s is a prescribed feature magnitude that serves a similar role to HFN. When $t = 0$, SFN reduces to FN-free learning. When $t = +\infty$, SFN reduces to HFN. Therefore, SFN can be viewed as an interpolation between FN-free learning and HFN. SFN has also been studied in [10].

SFN can make use of the instance-level information encoded in feature magnitude during training while still encouraging a feature normalization effects. Moreover, the difference between SFN and HFN can be viewed as using different optimization techniques to constrain the feature norm to a prescribed constant. HFN has the flavor of projected gradient descent where the solution will be projected to the feasible region to satisfy some constraint. In contrast, SFN is essentially a Lagrangian relaxation of the original problem where the feature norm is constrained. Therefore, their empirical performance could be quite different in practice, even if they share the same optimization target.

Dynamic feature magnitude. In contrast to HFN that uses a static feature magnitude, both FN-free learning and SFN can be viewed as a dynamic (data-dependent) way to control the feature magnitude. Moreover, there exist many other strategies that can dynamically control the feature magnitude to improve the empirical performance, such as [36].

4.3 Discussions and Open Problems

Optimal design of characteristic function. It is clear that the characteristic function is the key to large angular margin, but is there an optimal characteristic function? The answer to this question remains open. We argue that the optimal design of characteristic function should be dynamic and depends on the specific dataset, the network architecture, the optimizer, the stage of training (*i.e.*, the weights of the network), etc. Current studies on hyperspherical FR still focuses on a static characteristic function. [70], [71] explore an automatic way to learn a characteristic function from data, but those learned characteristic functions are still static ones and do not lead to a significant performance gain. [45] combines the sample quality to hyperspherical FR through a customized characteristic function that is dependent on the feature magnitude. How to design or learn a better characteristic function that is dynamically dependent on the data and also easy to optimize remains a huge challenge. Moreover, the underlying mechanism that determines the performance of a characteristic function stays a mystery and needs to be understood both empirically and theoretically.

Making better use of feature magnitude. In this paper, we have not considered to incorporate feature magnitude to testing and still stick to the cosine similarity for comparing pairs. However, it remains an interesting open problem whether it will be more beneficial to combine feature magnitude back to the similarity score (especially FN-free learning or SFN is used). We consider a generalized form of similarity score as $\mathcal{S}(\boldsymbol{x}_1, \boldsymbol{x}_2) = g(\|\boldsymbol{x}_1\|, \|\boldsymbol{x}_2\|) \cdot \cos(\theta_{1,2})$ where $\boldsymbol{x}_1, \boldsymbol{x}_2$ are deep features of two input samples, $\theta_{1,2}$ is the angle between $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, and $g(\|\boldsymbol{x}_1\|, \|\boldsymbol{x}_2\|)$ denotes a function with the norm of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ as input. We may require the function $g(\|\boldsymbol{x}_1\|, \|\boldsymbol{x}_2\|)$ to have a few properties: (i) permutation invariance: $g(\|\boldsymbol{x}_1\|, \|\boldsymbol{x}_2\|) = g(\|\boldsymbol{x}_2\|, \|\boldsymbol{x}_1\|)$ and (ii) adjustable magnitude augmentation. As a concrete example, we could use $g(\|\boldsymbol{x}_1\|, \|\boldsymbol{x}_2\|) = \|\boldsymbol{x}_1\|^t \cdot \|\boldsymbol{x}_2\|^t$ where t adjusts the augmentation strength of feature magnitude. $g(\|\boldsymbol{x}_1\|, \|\boldsymbol{x}_2\|) = 1$ reduces to the cosine similarity score. In general, how to design a good g is not clear and remains to be explored in future endeavours.

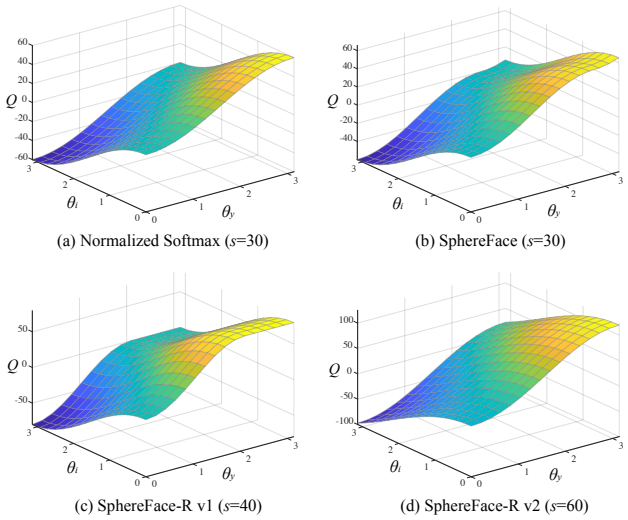


Fig. 5. A comparison of loss characteristics among normalized softmax, SphereFace, SphereFace-R v1 and SphereFace-R v2.

5 A Unified Characterization of Loss Functions in Hyperspherical Face Recognition

In this section, we take a closer look at what characterizes hyperspherical face recognition. As our unified framework in Section 3 discusses, a feature normalization strategy, a (non-)target angular function and a characteristic function can fully determine the loss function of a hyperspherical FR method. Particularly, the characteristic function $\Delta(\cdot)$ controls the property of the induced angular margin (*e.g.*, size, training stability). It does not consider the feature magnitude and only focuses on the difference between target and non-target function. Here we take a step further by showing a unified way to characterize the loss function as a whole. Specifically, we have the following general form of the loss function for hyperspherical FR:

$$\begin{aligned}\mathcal{L}_s &= \log\left(1 + \sum_{i \neq y} \exp\Big(\underbrace{s \cdot (\eta(\theta_i) - \eta(\theta_y) + \Delta(\theta_y))}_{:= Q(\theta_y, \theta_i, s, m)}\Big)\right) \\ &= \log\left(1 + \sum_{i \neq y} \exp\left(Q(\theta_y, \theta_i, s, m)\right)\right)\end{aligned} \tag{19}$$

where we define $Q(\theta_y, \theta_i, s, m)$ as the loss characteristics that fully determine how the loss function behaves. We compare the loss characteristics among normalized softmax [9], SphereFace and two variants of SphereFace-R in Fig. 5.

Although we show that the loss characteristics can fully determine the loss function, the underlying mechanisms of how the loss characteristics can affect the performance are largely unclear and remain to be investigated. Typically, s and m jointly specify the loss characteristics, and their roles could be partially coupled, which is also empirically observed in our ablation study.

From a back-propagation perspective, we have the gradient of $\mathcal{L}_s$ (*w.r.t.* either $\boldsymbol{x}$ or $\boldsymbol{W}_i, \forall i$) as

$$\mathcal{L}'_s = \sum_{i \neq y} \frac{\exp(Q(\theta_y, \theta_i, s, m))}{1 + \sum_{i \neq y} \exp(Q(\theta_y, \theta_i, s, m))} \cdot Q'(\theta_y, \theta_i, s, m) \quad (20)$$

where $Q'(\theta_y, \theta_i, s, m)$ denotes the gradient of loss characteristics. Quite interestingly, if we apply CGD to Eq. (19), then $Q'(\theta_y, \theta_i, s, m)$ for all hyperspherical FR methods will immediately become identical to that of the normalized softmax loss [9]. The only critical difference lies in the weighting factor $\rho_i = \frac{\exp(Q(\theta_y, \theta_i, s, m))}{1 + \sum_{i \neq y} \exp(Q(\theta_y, \theta_i, s, m))}$ in Eq. (20), where $Q(\theta_y, \theta_i, s, m)$ varies for different loss functions in the forward pass. This finding suggests that once CGD is applied, the gradient of every loss function in hyperspherical FR can be viewed as a particular weighting strategy to combine $Q'(\theta_y, \theta_i, s, m)$ of different $i \neq y$. In other words, only the weighting factors $\rho_i, \forall i$ in the gradient $\mathcal{L}'_s = \sum_{i \neq y} \rho_i \cdot Q'(\theta_y, \theta_i, s, m)$ will differ for different loss functions. Therefore, the design space for loss functions can be switched from finding $Q(\theta_y, \theta_i, s, m)$ to finding a weighting strategy for combining the gradients $Q'(\theta_y, \theta_i, s, m), \forall i \neq y$. Such a gradient weighting perspective reveals that searching for suitable m and m is equivalent to designing a good gradient weighting strategy. This may open a brand new gate to gain deeper understandings towards hyperspherical FR. Moreover, we believe that our novel loss characterization reformulation in Eq. (19) and Eq. (20) may inspire more effective designs for loss functions in hyperspherical FR.

6 Experiments and Results

In this section, we present comprehensive experiments to explore the properties of the SphereFace family. Experimental setup is introduced in Section 6.1. We perform ablation studies in Section 6.2 and Section 6.3 to investigate different variants of SphereFace and their hyperparameters. In Section 6.4, we evaluate our methods on the large-scale benchmarks and compare to ohter state-of-art methods.

6.1 Implementation Details

Preprocessing. Each face image is cropped based on the five face landmarks (*i.e.*, left eye, right eye, nasal tip, left mouth corner, and right mouth corner) detected by MTCNN [73] and RetinaFace [74] using similarity transformation. The size of the cropped image is set to 112×112, and each RGB pixel ($[0, 255]$) is normalized to $[-1, 1]$.

CNNs. The SphereFace Networks (SFNets) that are initially proposed in [2] are used as the backbone in our experiments. Slightly different from [2], we equip SFNets with batch normalization (BN) [75] to facilitate the model optimization. For better comparison to existing methods, we also evaluate our models with IResNet-100 [5] which is a 100-layer modified ResNet. The affine parameters in the last BN layer are enabled when NFN and SFN are used. We use SFNet-20 and SFNet-64 in the ablation and exploration, while SFNet-64 and IResNet-100 are adopted in large-scale benchmarks to achieve state-of-the-art performance.

TABLE 2
Statistics for the used datasets.

Dataset	# of ID	# of images	Split
VGGFace2 [76]	8.6K	3.1M	train
MS-Celeb-1M [77]	86K	5.8M	train
LFW [78]	5,749	13,233	validation
AgeDB-30 [79]	568	16,488	validation
CALFW [80]	5,749	11,652	validation
CPLFW [81]	5,749	12,174	validation
CFP [82]	500	7,000	validation
VGGFace2_test [76]	500	173k	validation
IJB-B [83]	1,845	76.8K	test
IJB-C [84]	3,531	148.8K	test
MegaFace (probe) [85]	530	3,530	test
MegaFace (distractor) [86]	690K	1M	test

TABLE 3
Varying m for no feature normalization on VGGFace2 (%).

m	SphereFace	SphereFace-R v1	SphereFace-R v2
1.1	53.09	55.18	52.71
1.2	**55.36**	**55.97**	**56.08**
1.3	55.32	51.11	50.19
1.4	44.95	43.04	37.78
1.5	34.23	31.54	30.94

Training. The training images are horizontally flipped for data augmentation. We train all the models on two popular training dataset: VGGFace2 [76] (3.1M images from 8.6K IDs) and MS-Celeb-1M (5.8M images from 86K IDs, also called MS1M-V2, the cleaned version of MS-Celeb-1M used in [5]). Detailed statistics of these datasets are given in Table 2. In our experiments, all the models are optimized using stochastic gradient descent with momentum 0.9. For VGGFace2, we train on 2 GPUs for 80k iterations, with a learning rate of 0.1 which is decreased by 10 at the 40k and 60k iteration. For MS-Celeb-1M, we train on 4 GPUs for 240k iterations. The learning rate is initialized as 0.1 and decreased by 10 at the iteration of 100k, 180k, and 220k.

Testing. We strictly follow the specific protocol provided in each dataset for evaluation. Table 2 shows the statistics of the testing sets. Given a face image, we extract two 512-dimensional embeddings from the original image and its horizontally flipped version, respectively. The final embedding is obtained by averaging the two. The scoring method is cosine similarity. The nearest neighbor classifier and thresholding are used for face identification and verification, respectively. To reduce the randomness, 5 models from the last 10k iterations will be used in testing and their averaged results are reported. Specifically, we evaluate the models at the iteration of 72k, 74k, 76k, 78k and 80k for VGGFace2, and 232k, 234k, 236k, 238k and 240k for MS-Celeb-1M.

6.2 Ablation and Exploration on VGGFace2

The validation set is a combination of multiple datasets, including LFW, AgeDB-30, CALFW, CPLFW, CFP-FP, CFP-

TABLE 4
Grid searching for m and s with hard feature normalization on VGGFace2. Results are in % and higher number indicates better performance.

m \ s	20	30	40	50	60
1.1	51.97	53.28	55.50	53.78	52.06
1.2	**53.16**	**61.37**	57.22	54.85	56.12
1.3	48.28	60.08	**60.89**	**60.37**	**57.09**
1.4	42.32	58.63	59.30	57.39	53.94
1.5	37.20	58.04	59.35	54.74	53.43
1.6	38.62	47.81	53.53	49.77	53.30

(a) Varying m and s for SphereFace.

m \ s	20	30	40	50	60
1.1	**50.88**	57.10	54.12	51.84	53.37
1.2	49.68	57.21	56.30	51.99	52.17
1.3	45.35	**59.15**	56.23	56.92	55.02
1.4	45.83	53.38	58.05	55.79	56.15
1.5	35.91	55.00	**60.45**	**58.29**	**58.95**
1.6	33.40	47.70	53.75	52.28	50.92

(b) Varying m and s for SphereFace-R v1.

m \ s	30	40	50	60	70
1.1	54.62	54.69	50.52	52.56	51.33
1.2	**55.59**	**60.75**	58.30	54.55	56.88
1.3	44.12	55.67	**61.07**	57.34	57.15
1.4	35.01	48.42	59.31	**62.72**	**58.12**
1.5	32.71	42.50	46.98	55.54	56.81
1.6	25.90	30.83	42.88	49.83	51.40

(c) Varying m and s for SphereFace-R v2.

TABLE 5
Varying t for soft feature normalization on VGGFace2 (%).

t	SphereFace ($m = 1.2, s = 30$)	SphereFace-R v1 ($m = 1.5, s = 40$)	SphereFace-R v2 ($m = 1.4, s = 60$)
0.05	57.08	39.43	55.53
0.1	**62.40**	43.96	60.61
0.2	61.24	54.40	60.30
0.5	58.68	**61.37**	**61.21**
1.0	57.10	60.59	58.34

TABLE 6
Ablation of CGD for different FN strategies on VGGFace2 (%).

FN	CGD	SphereFace ($m = 1.2, s = 30$)	SphereFace-R v1 ($m = 1.5, s = 40$)	SphereFace-R v2 ($m = 1.4, s = 60$)
NFN	✗	47.91	49.92	49.41
	✓	**55.36**	**55.97**	**56.08**
HFN	✗	46.02	28.78	36.61
	✓	**61.37**	**60.45**	**62.72**
SFN	✗	47.81	25.26	40.52
	✓	**62.40**	**61.37**	**61.21**

FF, and VGG2-FP. The statistics of these datasets are summarized in Table 2. In total, there are 43,000 testing pairs (21,500 positive pairs, and 21,500 negative pairs). The performance of a model is measured by the area under the ROC curve (AUC). Since it is important for a face recognition system to avoid false positives, we use AUC-x [87] as the metric, which integrates up to false positive rate of x ($x \in [0, 1]$). We find that $x = 0.0005$ achieves the best trade-off between stability and effectiveness. Since VGGFace2 is a relatively small training set (8.6K subjects [76]), we use the SFNet-20 model as the backbone in this section.

6.2.1 No Feature Normalization

We start by exploring SphereFace and SphereFace-R without feature normalization. Since there is only one effective hyperparameter (*i.e.*, the margin m), it is easy to find the optimal setting. We compare SphereFace, SphereFace-R v1 and SphereFace-R v2, and they represent different types of margins as shown in Table 1. Note that SphereFace with NFN is the same as the original SphereFace [2] (with additional CGD). Our methods are equivalent to the standard softmax cross-entropy loss when $m = 1.0$. We vary m from 1.1 to 1.5 and report the corresponding AUC-0.0005 in Table 3. It can be observed that small margin (*e.g.*, 1.1) results in inferior performance, because the learned features are not sufficiently discriminative. On the other hand, incorporating a margin that is too large can not produce good results as well, due to the increased difficulty in optimization. In our experiments, $m = 1.2$ achieves the best trade-off between feature discriminativeness and optimization difficulty, leading to the best performance for all types of margins.

TABLE 7
Varying m with no feature normalization on MS-Celeb-1M (%).

m	SphereFace	SphereFace-R v1	SphereFace-R v2
1.0	76.50	75.12	78.98
1.1	**79.89**	82.52	79.94
1.2	49.64	**84.66**	**84.78**
1.3	-	0.33	69.95

TABLE 8
Varying m with hard feature normalization on MS-Celeb-1M (%).

m	SphereFace		SphereFace-R v1		SphereFace-R v2	
	$s = 32$	$s = 64$	$s = 32$	$s = 64$	$s = 64$	$s = 96$
1.4	87.32	-	89.42	-	86.88	-
1.5	88.82	-	90.19	-	**91.58**	-
1.6	88.97	89.15	**90.78**	86.24	88.74	88.48
1.7	**91.03**	89.30	89.79	88.89	88.80	89.36
1.8	89.83	**90.53**	88.53	88.89	88.54	88.98
1.9	-	89.58	-	**89.61**	-	**89.71**
2.0	-	88.50	-	87.52	-	89.01

TABLE 9
Varying t with soft feature normalization on MS-Celeb-1M (%).

t	SphereFace ($m = 1.7, s = 32$)	SphereFace-R v1 ($m = 1.6, s = 32$)	t	SphereFace-R v2 ($m = 1.5, s = 64$)
0.6	54.30	81.27	0.1	78.18
0.8	85.40	81.24	0.2	**86.10**
1.0	**87.70**	**87.19**	0.4	83.34
1.2	85.00	87.15	0.6	85.50
1.4	78.52	83.67	0.8	81.75

6.2.2 Hard Feature Normalization

HFN introduces an additional hyperparameter s to the loss function, which controls the norm of the deep features. To show how m and s affect the performance, we perform a grid search by varying these two hyperparameters for all three types of margins including SphereFace, SphereFace-R v1 and SphereFace-R v2. The results are given in Table 4.

We have several observations. First, there usually exists an optimal margin hyperparameter m that leads to the best performance for each scale s. If m is smaller than the optimal value, the performance is usually improved as m increases. If m is greater than the optimal value, the performance is usually decreased as m increases. Second, for larger s, the corresponding optimal m will also tend to

TABLE 10
Evaluation on MegaFace, IJB-B and IJB-C. We use SFNet-20 as the backbone architecture and VGGFace2 as the training set for all the compared methods. Results are in % and higher number indicates better performance.

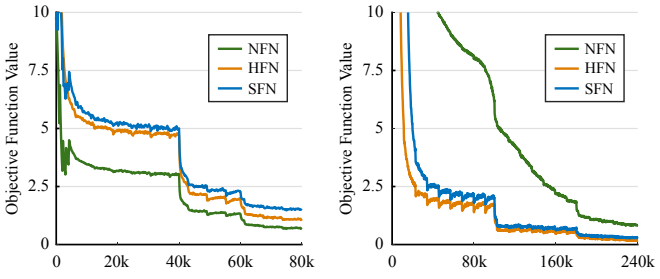
					MegaFace (refined)		IJB-B						IJB-C					
							1:1 Veri. TAR @ FAR			1:N Iden. TPIR @ FPIR			1:1 Veri. TAR @ FAR			1:N Iden. TPIR @ FPIR		
Method	FN	m	s	t	Iden.	Veri.	1e-6	1e-5	1e-4	top 1	1e-2	1e-1	1e-6	1e-5	1e-4	top 1	1e-2	1e-1
NormFace [9]	HFN	0.35	40	-	76.81	82.18	32.53	68.20	82.24	91.17	58.85	78.99	65.64	76.31	86.15	92.09	70.60	81.43
CosFace [3], [4]	HFN	0.35	40	-	81.38	85.73	**40.77**	73.66	85.51	91.96	67.97	82.77	70.43	80.21	88.75	93.09	75.36	84.90
ArcFace [5]	HFN	0.4	40	-	83.80	87.98	40.15	**76.52**	**87.50**	**92.26**	**70.25**	**85.02**	**74.32**	**82.49**	90.17	**93.79**	**78.22**	**86.71**
Circle Loss [26]	HFN	0.25	256	-	83.00	86.28	36.56	72.81	86.51	91.41	65.58	83.73	69.69	80.66	89.67	92.96	75.41	85.63
CurricularFace [40]	HFN	0.25	40	-	**88.32**	**91.82**	22.16	63.35	88.23	92.66	47.59	84.93	35.54	76.49	**91.10**	93.73	54.13	85.77
SphereFace [2]	NFN	1.2	-	-	85.55	90.03	**40.52**	**74.89**	**86.81**	92.86	**67.79**	**84.19**	**71.95**	81.46	**89.69**	**94.20**	76.20	85.85
SphereFace-R v1	NFN	1.2	-	-	**85.63**	**90.70**	35.42	73.87	86.59	**92.92**	66.75	83.95	71.60	**81.80**	89.67	94.12	**76.29**	**85.92**
SphereFace-R v2	NFN	1.2	-	-	85.02	89.24	39.17	73.80	86.36	92.85	67.60	83.66	70.96	80.61	89.32	93.95	75.30	85.04
SphereFace	HFN	1.2	30	-	**85.44**	**90.12**	**40.11**	75.44	87.43	**92.97**	67.70	84.87	73.79	83.02	90.37	**94.19**	78.18	86.90
SphereFace-R v1	HFN	1.5	40	-	83.39	87.42	39.39	75.78	86.82	92.53	68.92	84.43	72.39	82.33	89.83	93.85	77.96	86.39
SphereFace-R v2	HFN	1.4	60	-	85.15	89.35	36.68	**76.61**	**87.81**	92.85	**70.21**	**85.50**	**74.94**	**83.51**	**90.45**	94.06	**78.80**	**87.24**
SphereFace	SFN	1.2	30	0.1	86.06	90.59	39.55	76.00	87.63	93.14	68.37	85.21	74.59	83.34	90.59	94.27	78.93	87.12
SphereFace-R v1	SFN	1.5	40	0.5	**89.17**	**91.95**	42.38	**80.24**	**88.88**	**93.45**	**73.55**	**87.24**	**77.61**	**85.93**	**91.59**	**94.65**	**82.09**	**89.01**
SphereFace-R v2	SFN	1.4	60	0.5	89.10	91.75	**44.20**	78.91	88.86	93.12	70.84	87.15	75.91	85.63	91.38	94.31	81.04	88.54

be larger. For example, in Table 4(b), the optimal margin m is monotonously increased from 1.1 to 1.5 for the scale s ranging from 20 to 60. The performance is affected by both s and m in a coupled manner. The same pattern also appears in Table 4(a) and (c). Third, a wide range of s could lead to a satisfying performance, as long as m is properly tuned. For example, in Table 4(a), the corresponding AUC-0.0005 remains a relatively high value ($[57.09, 61.37]$) while the scale s varies from 30 to 60. We note that these observations are consistent across all types of multiplicative margin.

Hyperparameter tuning strategy. With the aforementioned observations, we summarize a simple yet effective strategy to search suitable hyperparameters for HFN. First, we uniformly pick the scale parameter s with a relatively large gap, *e.g.*, 20 or 30. The approximate range of s can follow the common setting in many existing approaches [2], [5]. Second, we find the optimal margin parameter m for each s with a uniform search. It has been shown in Table 4 that the range of m depends on the specific s we use. For larger s, its optimal m is typically larger as well. As a useful practice, we should gradually search larger m, as s increases. Finally, we choose the scale s and the margin m that lead to the best performance on the validation set. We find that such a simple hyperparameter searching is generally useful and can be applied to tuning different kinds of hyperspherical FR methods in practice.

6.2.3 Soft Feature Normalization

SFN is a soft regularization to constrain the feature norm and can be considered as a trade-off between NFN and HFN. SFN has a weighting hyperparameter t controlling the contribution of feature norm regularization term. Since there are three hyper-parameters, *i.e.*, m, s, and t, it is time-consuming and infeasible to enumerate all possible combinations of them. To efficiently evaluate SFN, we adopt the combination of m and s that leads to the best performance in Section 6.2.2 and fix them throughout the SFN experiment so that we can focus on how t will affect the performance.



(a) Training SFNet-20 on VGGFace2 (b) Training SFNet-64 on MS-Celeb-1M

Fig. 6. Training objective of SphereFace-R v2 with NFN, HFN and SFN on (a) VGGFace2 and (b) MS-Celeb-1M. For SFN, we only plot the softmax-based loss and the feature norm regularization is not plotted.

We give the AUC-0.0005 in Table 5. SphereFace achieves 62.4% with SFN and 61.37% with HFN. SphereFace-R v1 achieves 61.37% with SFN and 60.45% with HFN. SphereFace-R v2 achieves 61.21% with SFN and 62.72% with HFN. Our experiments show that that SFN generally yields comparable or even better results than HFN with a properly chosen t when our models are trained on VGGFace2.

6.2.4 Characteristic Gradient Detachment

In Table 6, we compare the models trained with or without CGD. Except the usage of CGD, the experiments are performed with exactly the same experimental settings (*e.g.*, dataset, architecture, training setup etc). We have evaluated CGD on all three FN strategies (*i.e.*, NFN, HFN and SFN). The results show that CGD significantly improves the results in all scenarios, which validates the importance of simplifying the gradient of the characteristic function. CGD works particularly well for SphereFace-R v1, since it enables the gradient propagation for large margin ($m > \frac{\pi}{\theta}$ in Eq. 11). Here we use the best-performing hyperparameters from our previous experiments. In fact, the consistent improvements can also be obtained from CGD with the other

TABLE 11
Evaluation on MegaFace, IJB-B and IJB-C. We use SFNet-64 as the backbone architecture and MS-Celeb-1M as the training set for all the compared methods. Results are in % and higher number indicates better performance.

					MegaFace (refined)		IJB-B						IJB-C					
							1:1 Veri. TAR @ FAR			1:N Iden. TPIR @ FPIR			1:1 Veri. TAR @ FAR			1:N Iden. TPIR @ FPIR		
Method	FN	m	s	t	Iden.	Veri.	1e-6	1e-5	1e-4	top 1	1e-2	1e-1	1e-6	1e-5	1e-4	top 1	1e-2	1e-1
NormFace [9]	HFN	-	30	-	89.24	90.76	40.56	75.30	90.22	92.49	64.62	88.19	70.17	85.88	92.69	93.70	77.97	89.81
CosFace [3], [4]	HFN	0.35	64	-	98.05	98.45	37.82	82.99	94.20	94.69	70.61	93.03	78.01	92.29	95.87	95.91	84.59	94.53
ArcFace [5]	HFN	0.5	64	-	**98.45**	98.39	41.02	**86.16**	**94.82**	**94.88**	**77.92**	**93.79**	**84.47**	**93.25**	**96.25**	**96.12**	**88.80**	**95.08**
Circle Loss [26]	HFN	0.25	256	-	98.29	**98.67**	41.65	82.76	94.09	94.64	74.63	92.83	81.18	91.59	95.83	95.77	84.56	94.15
CurricularFace [40]	HFN	0.5	64	-	98.43	98.62	**43.76**	85.55	94.61	94.82	76.01	93.37	83.35	92.95	96.11	96.04	87.88	94.76
SphereFace [2]	NFN	1.1	-	-	93.04	94.37	37.78	69.99	88.70	91.98	60.43	86.28	62.10	83.03	91.74	93.30	71.45	88.11
SphereFace-R v1	NFN	1.2	-	-	93.75	94.80	44.83	**83.07**	92.12	**93.56**	70.59	90.80	**77.72**	89.78	94.14	**94.94**	84.55	**92.23**
SphereFace-R v2	NFN	1.2	-	-	**94.74**	**95.51**	**47.82**	82.82	**92.15**	93.51	**72.38**	**90.91**	76.14	**89.85**	**94.17**	**94.94**	**84.58**	92.17
SphereFace	HFN	1.7	32	-	**98.16**	98.46	48.83	86.66	**94.36**	94.84	76.35	93.20	83.57	92.79	**95.82**	**96.07**	87.74	94.47
SphereFace-R v1	HFN	1.6	32	-	98.03	98.30	**48.84**	**88.16**	94.31	**94.98**	**79.18**	**93.23**	**85.65**	**93.17**	95.72	**96.07**	**89.94**	**94.55**
SphereFace-R v2	HFN	1.5	64	-	98.04	**98.48**	45.77	86.52	94.08	94.70	74.13	93.13	82.07	92.61	95.63	95.92	88.00	94.32
SphereFace	SFN	1.7	32	1.0	97.84	**98.28**	**44.42**	85.89	**94.13**	**94.84**	77.37	93.14	**83.05**	92.25	**95.67**	95.95	87.72	94.19
SphereFace-R v1	SFN	1.6	32	1.0	97.42	97.79	44.24	84.95	93.70	94.58	**77.66**	92.66	81.92	91.10	95.27	95.65	85.70	93.73
SphereFace-R v2	SFN	1.5	64	0.2	**97.93**	98.19	40.06	**86.37**	94.12	94.78	76.94	**93.21**	82.94	**92.29**	95.62	**95.99**	**87.83**	**94.42**

hyperparameters. We observe that CGD can effectively improve SphereFace and SphereFace-R under all FN strategies. Because of the consistent effectiveness of CGD, we use CGD for both SphereFace and SphereFace-R by default and the results in the other sections are also obtained with CGD.

6.3 Ablation and Exploration on MS-Celeb-1M

In this section, we conduct ablation and exploration with a deeper network that is trained on a large-scale dataset. Specifically, we use SFNet-64 as the backbone network architecture and MS-Celeb-1M as the training set. We report AUC-0.0005 on the same validation set as Section 6.2.

6.3.1 No Feature Normalization

We evaluate SphereFace, SphereFace-R v1 and SphereFace-R v2 without feature normalization. Note that SphereFace with NFN is equivalent to the original SphereFace [2] with CGD. The results are given in Table 7. Similar to the experiments on VGGFace2, NFN generally works well with small margins, achieving 79.89%, 84.66%, and 84.78% with m being 1.1, 1.2, and 1.2, respectively. The increased number of training identities (from 8K to 86K) leads to severe optimization difficulty during training. This can be empirically observed from the smaller optimal m from SphereFace and the dramatically decreased performance from both SphereFace and SphereFace-R with larger margins.

6.3.2 Hard Feature Normalization

Following our hyperparameter tuning strategy given in Section 6.2.2, we search the optimal combination of s and m. Since the performance is stable across a wide range of s, here we use a relatively large gap (*i.e.*, 32). As mentioned in Section 6.2.2, the optimal m tends to increase with larger s. We search two feature scale hyperparameters for SphereFace (*i.e.*, $s = 32, 64$), SphereFace-R v1 (*i.e.*, $s = 32, 64$) and SphereFace-R v2 (*i.e.*, $s = 64, 128$). Based on our observation on VGGFace2, the optimal m for larger s also tends to be larger. Therefore for smaller s, we search m from 1.4 to 1.8, and for larger s, we search m from 1.6 to 2.0. The results in Table 8 well match our expectation that there is only one optimal m for each s and the optimal m will increase with larger s. Under the same s, we discover that as m deviates from its optimal value, the performance will also become worse. The distribution of the performance for different m exhibits a strong unimodality, which can largely benefit the hyperparameter tuning. The best models of SphereFace, SphereFace-R v1 and SphereFace-R v2 achieve 91.03%, 90.78% and 91.58% with $s = 32$, 32, and 64, respectively. The performance on MS-Celeb-1M is also much better than that on VGGFace2, which shows that our methods can easily enjoy the accuracy boost from larger training set and deeper network. Moreover, the performance of both SphereFace and SphereFace-R is not very sensitive to m and remains stable for a wide range of m.

6.3.3 Soft Feature Normalization

Similar to Section 6.2.3, we adopt the best-performing settings (m and s) from HFN and vary the hyperparameter t in order to evaluate the performance of SFN. The results are reported in Table 9. With a properly tuned t, SFN achieves 87.70%, 87.19% and 86.10% AUC-0.0005 for SphereFace, SphereFace-R v1 and SphereFace-R v2, respectively. We observe that the performance of SFN on MS-Celeb-1M are not as good as HFN, which contradicts the observation on VGGFace2. This implies that SFN may be sensitive to the distribution of the training data. We hypothesize that SFN is more sensitive to noisy samples (since MS-Celeb-1M has much more low-quality and noisy images than VGGFace2).

6.4 Evaluation on Large-scale Benchmarks

6.4.1 Experiments with SFNet-20 and SFNet-64

In this section, we evaluate the performance of both SphereFace and SphereFace-R with three FN strategies on popular large-scale benchmarks (*i.e.*, IJB-B, IJB-C and MegaFace). We also provide fair comparisons to state-of-the-art methods (with the same training set and backbone

TABLE 12
Evaluation on MegaFace, IJB-B and IJB-C. We use IResNet-100 as the backbone architecture and MS-Celeb-1M as the training set. Results are in % and higher number indicates better performance. For ArcFace, we present results from their paper and our re-implementation.

					MegaFace (refined)		IJB-B						IJB-C					
							1:1 Veri. TAR @ FAR			1:N Iden. TPIR @ FPIR			1:1 Veri. TAR @ FAR			1:N Iden. TPIR @ FPIR		
Method	FN	m	s	t	Iden.	Veri.	1e-6	1e-5	1e-4	top 1	1e-2	1e-1	1e-6	1e-5	1e-4	top 1	1e-2	1e-1
CosFace [3], [4]	HFN	0.4	64	-	**98.70**	**98.87**	**43.67**	88.83	**95.23**	**95.35**	80.50	**94.49**	85.29	94.33	**96.62**	**96.53**	90.69	**95.61**
ArcFace (results in [5])	HFN	0.5	64	-	98.35	98.48	38.28	89.33	94.25	-	-	-	**86.25**	93.15	95.65	-	-	-
ArcFace [5]	HFN	0.5	64	-	98.67	98.46	43.43	**90.40**	95.02	95.14	**81.36**	94.26	86.00	**94.49**	96.39	96.47	**91.91**	95.51
SphereFace [2]	NFN	1.1	-	-	94.95	95.76	**43.02**	73.79	90.19	92.67	64.09	87.80	68.83	85.77	92.82	93.89	76.83	89.93
SphereFace-R v1	NFN	1.2	-	-	96.49	97.22	38.01	81.31	92.58	94.15	70.49	91.20	77.94	90.23	94.83	95.34	**84.81**	92.91
SphereFace-R v2	NFN	1.2	-	-	**97.14**	**97.56**	41.73	**82.60**	**93.00**	**94.42**	**70.93**	**91.44**	**78.64**	**90.70**	**95.03**	**95.45**	84.75	**93.11**
SphereFace	HFN	1.7	32	-	**98.63**	**99.04**	47.33	**90.14**	94.87	95.13	82.57	**94.30**	**87.86**	**94.36**	**96.25**	**96.45**	**91.68**	**95.36**
SphereFace-R v1	HFN	1.6	32	-	98.61	98.61	**47.82**	89.06	**94.89**	**95.18**	**82.69**	94.09	86.30	93.91	96.21	96.38	91.16	95.19
SphereFace-R v2	HFN	1.5	64	-	98.46	98.69	41.22	88.86	94.77	95.18	79.57	93.78	84.21	93.55	96.14	96.27	89.85	94.95
SphereFace	SFN	1.7	32	1.0	98.23	98.50	45.35	84.86	94.20	94.90	76.38	93.32	80.83	92.17	95.70	95.91	86.44	94.37
SphereFace-R v1	SFN	1.6	32	1.0	98.07	98.15	42.97	**87.92**	94.32	94.94	**78.47**	93.32	**85.43**	**93.06**	95.83	96.02	**89.13**	94.56
SphereFace-R v2	SFN	1.5	64	0.2	**98.26**	**98.58**	**45.64**	86.55	**94.51**	**95.10**	76.53	**93.65**	80.19	93.01	**95.96**	**96.19**	87.39	**94.88**

network). We use the same models from Section 6.2 and Section 6.3 with the best-performing hyperparameters on the validation set. Specifically, we train SFNet-20 on VGGFace2 and SFNet-64 on MS-Celeb-1M. We compare our models to current state-of-the-art methods, *i.e.*, NormFace [9], CosFace [3], [4], ArcFace [5], circle loss [26] and CurricularFace [40]. The hyperparameters of these methods are tuned to achieve to the best validation performance.

We make several useful observations from Table 10 and Table 11. First, the results on large-scale testing sets are consistent with those on the validation set, especially the metrics at low false acceptance rate (FAR) or false positive identification rates (FPIR). The consistent performance demonstrates the effectiveness of the selected validation set and metric. This indicates that the models that achieve higher performance on the validation set usually show better results on MegaFace and IJB as well.

Second, different types of margins tend to achieve similar performance, while different types of FN strategies have significantly different results. In Fig. 6, we first show the training losses for different FN strategies. Fig. 6(a) shows that all the models equipped with NFN, HFN or SFN converge well on VGGFace2, a relatively small, clean and high-quality training set. From Table 10, NFN and HFN show comparable generalization ability, while SFN achieves the best performance among all FN strategies. This implies that making good use of the magnitude information during training can effectively improve the results. Fig. 6(b) shows that NFN is unable to converge to a sufficiently small training loss on MS-Celeb-1M, a large, noisy and low-quality dataset. From Table 11, we can observe that NFN indeed converges to a bad local minima that generalizes poorly. By introducing a magnitude regularization term to the objective function, SFN can effectively help the models escaping from the bad local minima. In contrast to the results on VGGFace2, we find that SFN performs worse than HFN on MS-Celeb-1M, implying that SFN may be more sensitive to noisy samples in the training set and HFN may be more robust to different training sets than NFN and SFN.

Finally, with our proposed modifications, all variants in our SphereFace family shows competitive results compared to the state-of-the-art methods. Both SphereFace and SphereFace-R perform particularly well under the low FAR, such as 1:1 verification TAR at 1e-6 and 1:N identification TPIR at 1e-1 FPIR. These metrics are very important in designing a robust face recognition system in practice.

6.4.2 *Experiments with IResNet-100*

In order to have a comprehensive comparison with the published results, we conduct the experiments to train our methods on MS-Celeb-1M [77] with IResNet-100 [5]. Comparing the results in Table 12 and Table 11, we observe that IResNet100 achieves better performance than those using SFNet-64 (with the same set of hyperparameters), establishing a higher baseline for SphereFace. For different FN strategies, IResNet100 performs similarly to SFNet64 in the sense that HFN is slightly better than SFN and they are both better than NFN. In general, both SphereFace and SphereFace-R are generally comparable to CosFace and ArcFace. SphereFace with HFN achieves the best performance on MegaFace. More interestingly, we find that both SphereFace and SphereFace-R v1 with HFN achieve significantly better 1:1 verification performance at low FAR than all the compared methods on IJB.

7 Concluding Remarks

Our paper proposes a novel framework that unifies hyperspherical face recognition. This framework provides a general principle for a loss function to incorporate large angular margins. Under this framework, we substantially extend and improve our previous work on SphereFace [2] by addressing training instability and significantly improving empirical performance. Specifically, we propose two new types of multiplicative margins that effectively implement the original intuition of SphereFace. Moreover, we also come up with a novel implementation technique called characteristic gradient detachment to further improve training stability and generalization. Extensive experiments on a number of popular benchmarks are conducted to validate the superiority of our SphereFace family.

Based on the unified framework, our paper demonstrates strong flexibility and many unique advantages of hyperspherical FR. There still exist a number of exciting yet under-explored open problems in hyperspherical FR, such as how to design better angular margin, how to effectively incorporate feature magnitude into training and testing, how to learn the loss function directly from data, etc. We also present a few useful characterizations for the loss function in hyperspherical FR, leading to multiple equivalent loss design spaces. Current popular loss functions only represent a very small and limited subset in the huge design space of hyperspherical FR. We expect that more work can be devoted to this promising line of research in the future.

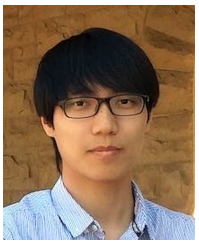
ACKNOWLEDGMENTS

The authors would like to sincerely thank Haoran Sun, Yuyu Zhang and Will Powell for generously helping us to schedule computing resources, and Hanchen Wang for proofreading. Weiyang Liu is supported by a Cambridge-Tübingen Fellowship, an NVIDIA GPU grant, DeepMind and the Leverhulme Trust via CFI. Adrian Weller acknowledges support from a Turing AI Fellowship under grant EP/V025379/1, The Alan Turing Institute under EPSRC grant EP/N510129/1 and TU/B/000074, and the Leverhulme Trust via CFI. This work is partially supported by the Defence Science and Technology Agency (DSTA), Singapore under contract number A025959, and this paper does not reflect the position or policy of DSTA and no official endorsement should be inferred.

REFERENCES

[1] W. Liu, Y. Wen, Z. Yu, and M. Yang, "Large-margin softmax loss for convolutional neural networks," in *ICML*, 2016.
[2] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "Sphereface: Deep hypersphere embedding for face recognition," in *CVPR*, 2017.
[3] F. Wang, W. Liu, H. Liu, and J. Cheng, "Additive margin softmax for face verification," *arXiv preprint arXiv:1801.05599*, 2018.
[4] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu, "Cosface: Large margin cosine loss for deep face recognition," in *CVPR*, 2018.
[5] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *CVPR*, 2019.
[6] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *CVPR*, 2014.
[7] Y. Sun, X. Wang, and X. Tang, "Deep learning face representation from predicting 10,000 classes," in *CVPR*, 2014.
[8] W. Liu, Z. Liu, Z. Yu, B. Dai, R. Lin, Y. Wang, J. M. Rehg, and L. Song, "Decoupled networks," in *CVPR*, 2018.
[9] F. Wang, X. Xiang, J. Cheng, and A. L. Yuille, "Normface: L2 hypersphere embedding for face verification," in *ACM-MM*, 2017.
[10] Y. Zheng, D. K. Pal, and M. Savvides, "Ring loss: Convex feature normalization for face recognition," in *CVPR*, 2018.
[11] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *BMVC*, 2015.
[12] Y. Sun, Y. Chen, X. Wang, and X. Tang, "Deep learning face representation by joint identification-verification," in *NIPS*, 2014.
[13] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *CVPR*, 2015.
[14] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *ECCV*, 2016.
[15] Y. Sun, X. Wang, and X. Tang, "Sparsifying neural network connections for face recognition," in *CVPR*, 2016.
[16] J. Liu, Y. Deng, and C. Huang, "Targeting ultimate accuracy: Face recognition via deep embedding," *arXiv preprint:1506.07310*, 2015.
[17] J. Hu, J. Lu, and Y.-P. Tan, "Discriminative deep metric learning for face verification in the wild," in *CVPR*, 2014.
[18] J. Lu, G. Wang, W. Deng, P. Moulin, and J. Zhou, "Multi-manifold deep metric learning for image set classification," in *CVPR*, 2015.
[19] H. O. Song, Y. Xiang, S. Jegelka, and S. Savarese, "Deep metric learning via lifted structured feature embedding," in *CVPR*, 2016.
[20] Y. Movshovitz-Attias, A. Toshev, T. K. Leung, S. Ioffe, and S. Singh, "No fuss distance metric learning using proxies," in *ICCV*, 2017.
[21] C.-Y. Wu, R. Manmatha, A. J. Smola, and P. Krahenbuhl, "Sampling matters in deep embedding learning," in *ICCV*, 2017.
[22] W. Ge, "Deep metric learning with hierarchical triplet loss," in *ECCV*, 2018.
[23] Y. Duan, W. Zheng, X. Lin, J. Lu, and J. Zhou, "Deep adversarial metric learning," in *CVPR*, 2018.
[24] Q. Qian, L. Shang, B. Sun, J. Hu, H. Li, and R. Jin, "Softtriple loss: Deep metric learning without triplet sampling," in *ICCV*, 2019.
[25] X. Wang, Y. Hua, E. Kodirov, G. Hu, R. Garnier, and N. M. Robertson, "Ranked list loss for deep metric learning," in *CVPR*, 2019.
[26] Y. Sun, C. Cheng, Y. Zhang, C. Zhang, L. Zheng, Z. Wang, and Y. Wei, "Circle loss: A unified perspective of pair similarity optimization," in *CVPR*, 2020.
[27] K. Musgrave, S. Belongie, and S.-N. Lim, "A metric learning reality check," in *ECCV*, 2020.
[28] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A comprehensive study on center loss for deep face recognition," *International Journal of Computer Vision*, vol. 127, no. 6, pp. 668–683, 2019.
[29] R. Ranjan, C. D. Castillo, and R. Chellappa, "L2-constrained softmax loss for discriminative face verification," *arXiv preprint arXiv:1703.09507*, 2017.
[30] Y. Liu, H. Li, and X. Wang, "Rethinking feature discrimination and polymerization for large-scale recognition," *arXiv preprint arXiv:1710.00870*, 2017.
[31] W. Liu, Y.-M. Zhang, X. Li, Z. Yu, B. Dai, T. Zhao, and L. Song, "Deep hyperspherical learning," in *NIPS*, 2017.
[32] W. Liu, R. Lin, Z. Liu, L. Liu, Z. Yu, B. Dai, and L. Song, "Learning towards minimum hyperspherical energy," in *NeurIPS*, 2018.
[33] R. Ranjan, A. Bansal, H. Xu, S. Sankaranarayanan, J.-C. Chen, C. D. Castillo, and R. Chellappa, "Crystal loss and quality pooling for unconstrained face verification and recognition," *arXiv preprint arXiv:1804.01159*, 2018.
[34] K. Zhao, J. Xu, and M.-M. Cheng, "Regularface: Deep face recognition via exclusive regularization," in *CVPR*, 2019.
[35] Y. Duan, J. Lu, and J. Zhou, "Uniformface: Learning deep equidistributed representation for face recognition," in *CVPR*, 2019.
[36] X. Zhang, R. Zhao, Y. Qiao, X. Wang, and H. Li, "Adacos: Adaptively scaling cosine logits for effectively learning deep face representations," in *CVPR*, 2019.
[37] H. Liu, X. Zhu, Z. Lei, and S. Z. Li, "Adaptiveface: Adaptive margin and sampling for face recognition," in *CVPR*, 2019.
[38] Y. Wu, Y. Wu, R. Gong, Y. Lv, K. Chen, D. Liang, X. Hu, X. Liu, and J. Yan, "Rotation consistent margin loss for efficient low-bit face recognition," in *CVPR*, 2020.
[39] I. Kim, S. Han, S.-J. Park, J.-W. Baek, J. Shin, J.-J. Han, and C. Choi, "Discface: Minimum discrepancy learning for deep face recognition," in *ACCV*, 2020.
[40] Y. Huang, Y. Wang, Y. Tai, X. Liu, P. Shen, S. Li, J. Li, and F. Huang, "Curricularface: adaptive curriculum learning loss for deep face recognition," in *CVPR*, 2020.
[41] J. Deng, J. Guo, T. Liu, M. Gong, and S. Zafeiriou, "Sub-center arcface: Boosting face recognition by large-scale noisy web faces," in *ECCV*, 2020.
[42] Y. Kim, W. Park, M.-C. Roh, and J. Shin, "Groupface: Learning latent groups and constructing group-based representations for face recognition," in *CVPR*, 2020.
[43] Y. Zhong, W. Deng, J. Hu, D. Zhao, X. Li, and D. Wen, "Sface: sigmoid-constrained hypersphere loss for robust face recognition," *IEEE Transactions on Image Processing*, vol. 30, pp. 2587–2598, 2021.
[44] S. Li, J. Xu, X. Xu, P. Shen, S. Li, and B. Hooi, "Spherical confidence learning for face recognition," in *CVPR*, 2021.
[45] Q. Meng, S. Zhao, Z. Huang, and F. Zhou, "Magface: A universal representation for face recognition and quality assessment," in *CVPR*, 2021.
[46] Y. Wen, W. Liu, A. Weller, B. Raj, and R. Singh, "Sphereface2: Binary classification is all you need for deep face recognition," in *ICLR*, 2022.

[47] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang, "A closer look at few-shot classification," *arXiv preprint arXiv:1904.04232*, 2019.
[48] W. Liu, Z. Liu, J. M. Rehg, and L. Song, "Neural similarity learning," in *NeurIPS*, 2019.
[49] P. Mettes, E. van der Pol, and C. Snoek, "Hyperspherical prototype networks," in *NeurIPS*, 2019.
[50] W. Liu, R. Lin, Z. Liu, L. Xiong, B. Schölkopf, and A. Weller, "Learning with hyperspherical uniformity," in *AISTATS*, 2021.
[51] W. Liu, R. Lin, Z. Liu, J. M. Rehg, L. Paull, L. Xiong, L. Song, and A. Weller, "Orthogonal over-parameterized training," in *CVPR*, 2021.
[52] B. Yu and D. Tao, "Deep metric learning with tuplet margin loss," in *ICCV*, 2019.
[53] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *CVPR*, 2020.
[54] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *ICML*, 2020.
[55] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Pires, Z. Guo, M. Azar *et al.*, "Bootstrap your own latent: A new approach to self-supervised learning," in *NeurIPS*, 2020.
[56] T. Wang and P. Isola, "Understanding contrastive representation learning through alignment and uniformity on the hypersphere," in *ICML*, 2020.
[57] S. W. Park and J. Kwon, "Sphere generative adversarial network based on geometric moment matching," in *CVPR*, 2019.
[58] T. R. Davidson, L. Falorsi, N. De Cao, T. Kipf, and J. M. Tomczak, "Hyperspherical variational auto-encoders," in *UAI*, 2018.
[59] T. S. Cohen, M. Geiger, J. Köhler, and M. Welling, "Spherical cnns," in *ICLR*, 2018.
[60] Y. Rao, J. Lu, and J. Zhou, "Spherical fractal convolutional neural networks for point cloud recognition," in *CVPR*, 2019.
[61] R. Lin, W. Liu, Z. Liu, C. Feng, Z. Yu, J. M. Rehg, L. Xiong, and L. Song, "Regularizing neural networks via minimizing hyperspherical energy," in *CVPR*, 2020.
[62] X. Fan, W. Jiang, H. Luo, and M. Fei, "Spherereid: Deep hypersphere manifold embedding for person re-identification," *Journal of Visual Communication and Image Representation*, vol. 60, pp. 51–58, 2019.
[63] H.-X. Yu, W.-S. Zheng, A. Wu, X. Guo, S. Gong, and J.-H. Lai, "Unsupervised person re-identification by soft multilabel learning," in *CVPR*, 2019.
[64] Y. Hao, N. Wang, J. Li, and X. Gao, "Hsme: hypersphere manifold embedding for visible thermal person re-identification," in *AAAI*, 2019.
[65] M. Hajibabaei and D. Dai, "Unified hypersphere embedding for speaker recognition," *arXiv preprint arXiv:1807.08312*, 2018.
[66] Y. Liu, L. He, and J. Liu, "Large margin softmax loss for speaker verification," in *Interspeech*, 2019.
[67] R. Li, N. Li, D. Tuo, M. Yu, D. Su, and D. Yu, "Boundary discriminative large margin cosine loss for text-independent speaker verification," in *ICASSP*, 2019.
[68] Y. Fathullah, C. Zhang, and P. C. Woodland, "Improved large-margin softmax loss for speaker diarisation," in *ICASSP*, 2020.
[69] Y. Meng, J. Huang, G. Wang, C. Zhang, H. Zhuang, L. Kaplan, and J. Han, "Spherical text embedding," in *NeurIPS*, 2019.
[70] X. Wang, S. Wang, C. Chi, S. Zhang, and T. Mei, "Loss function search for face recognition," in *ICML*, 2020.
[71] C. Li, X. Yuan, C. Lin, M. Guo, W. Wu, J. Yan, and W. Ouyang, "Am-lfs: Automl for loss function search," in *ICCV*, 2019.
[72] B. Chen, W. Liu, Z. Yu, J. Kautz, A. Shrivastava, A. Garg, and A. Anandkumar, "Angular visual hardness," in *ICML*, 2020.
[73] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multi-task cascaded convolutional networks," *arXiv preprint:1604.02878*, 2016.
[74] J. Deng, J. Guo, E. Ververas, I. Kotsia, and S. Zafeiriou, "Retinaface: Single-shot multi-level face localisation in the wild," in *CVPR*, 2020.
[75] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015.
[76] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, "Vggface2: A dataset for recognising faces across pose and age," in *IEEE international conference on automatic face & gesture recognition*, 2018.
[77] Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao, "Ms-celeb-1m: A dataset and benchmark for large-scale face recognition," in *European conference on computer vision*. Springer, 2016, pp. 87–102.
[78] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments," Technical Report, Tech. Rep., 2007.
[79] S. Moschoglou, A. Papaioannou, C. Sagonas, J. Deng, I. Kotsia, and S. Zafeiriou, "Agedb: the first manually collected, in-the-wild age database," in *CVPR Workshops*, 2017.
[80] T. Zheng and W. Deng, "Cross-pose lfw: A database for studying cross-pose face recognition in unconstrained environments," *Technical Report*, 2018.
[81] T. Zheng, W. Deng, and J. Hu, "Cross-age lfw: A database for studying cross-age face recognition in unconstrained environments," *arXiv preprint arXiv:1708.08197*, 2017.
[82] S. Sengupta, J.-C. Chen, C. Castillo, V. M. Patel, R. Chellappa, and D. W. Jacobs, "Frontal to profile face verification in the wild," in *WACV*, 2016.
[83] C. Whitelam, E. Taborsky, A. Blanton, B. Maze, J. Adams, T. Miller, N. Kalka, A. K. Jain, J. A. Duncan, K. Allen *et al.*, "Iarpa janus benchmark-b face dataset," in *CVPR workshops*, 2017.
[84] B. Maze, J. Adams, J. A. Duncan, N. Kalka, T. Miller, C. Otto, A. K. Jain, W. T. Niggel, J. Anderson, J. Cheney *et al.*, "Iarpa janus benchmark-c: Face dataset and protocol," in *International Conference on Biometrics*, 2018.
[85] I. Kemelmacher-Shlizerman, S. M. Seitz, D. Miller, and E. Brossard, "The megaface benchmark: 1 million faces for recognition at scale," in *CVPR*, 2016.
[86] D. Miller, E. Brossard, S. Seitz, and I. Kemelmacher-Shlizerman, "Megaface: A million faces for recognition at scale," *arXiv preprint:1505.02108*, 2015.
[87] C. Barra, B. Alvarez, S. Paul, A. Sette, B. Peters, M. Andreatta, S. Buus, and M. Nielsen, "Footprints of antigen processing boost mhc class ii natural ligand predictions," *Genome medicine*, vol. 10, no. 1, pp. 1–15, 2018.

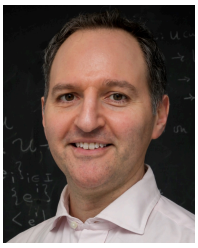


Weiyang Liu is currently conducting research at the University of Cambridge, UK and the Max Planck Institute for Intelligent Systems, Tübingen, Germany under the Cambridge-Tübingen Fellowship Program. Prior to joining this program, he has been with College of Computing, Georgia Institute of Technology, Atlanta, GA, USA. His research interests broadly lie in deep learning, representation learning, interactive machine learning and causality.

Yandong Wen received the B.S. and M.S. degrees from South China University of Technology, Guangzhou, China in 2013 and 2016, respectively. He is a Ph.D. candidate at Carnegie Mellon University, Pittsburgh, PA, USA, where he works with Bhiksha Raj and Rita Singh. His current research interests are deep learning for face recognition, audio-visual association learning, and 3D face reconstruction.

Bhiksha Raj (Fellow, IEEE) received the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in 2000. He is a Professor of the Computer Science Department, Carnegie Mellon University where he leads the Machine Learning for Signal Processing Group. He joined the Carnegie Mellon faculty in 2009, after spending time at the Compaq Cambridge Research Labs and Mitsubishi Electric Research Labs. He has devoted his career to developing speech- and audio-processing technology. He has had several seminal contributions in the areas of robust speech recognition, audio content analysis and signal enhancement, and has pioneered the area of privacy-preserving speech processing. He is also the Chief Architect of the popular Sphinx-4 speech-recognition system.

Rita Singh received the B.Sc.(Hons.) degree in physics and the M.Sc. degree in exploration geophysics, both from the Banaras Hindu University, India. She received the Ph.D degree in geophysics in 1996 from the National Geophysical Research Institute of the Council of Scientific and Industrial Research, India. She is currently a Member of the Research Faculty at the School of Computer Science, Carnegie Mellon University (CMU), Pittsburgh, PA, USA. From March 1996 to November 1997, she was a Postdoctoral Fellow with the Tata Institute of Fundamental Research, India, where she worked with the Condensed Matter Physics and Computer Systems and Communications Groups. During this period, she worked on nonlinear dynamical systems and signal processing as an extension of her doctoral work on nonlinear geodynamics and chaos. Since November 1997, she has been affiliated with the Robust Speech Recognition and SPHINX Groups at CMU. She currently works on core algorithmic aspects of computer voice recognition, and artificial intelligence applied to voice forensics. Her focus is on the development of technology for the automated discovery, measurement, representation and learning of the information encoded in voice signal for optimal voice intelligence.



Adrian Weller received his undergraduate degree from the University of Cambridge, and his PhD from Columbia University in New York. He is Programme Director for AI at The Alan Turing Institute, the UK national institute for data science and AI, where he is also a Turing Fellow leading work on safe and ethical AI. He is a Principal Research Fellow in Machine Learning at Cambridge, and at the Leverhulme Centre for the Future of Intelligence where he is Programme Director for Trust and Society. His interests span AI, its commercial applications and helping to ensure beneficial outcomes for society. He is Co-Director of the European Laboratory for Learning and Intelligent Systems (ELLIS) programme on Human-centric Machine Learning. Previously, he held senior roles in finance.