


## Article

# UnifiedFace: A Uniform Margin Loss Function for Face Recognition

Feng Zhao <sup>1</sup>, Peng Zhang <sup>1,\*</sup>, Ran Zhang <sup>2</sup> and Mengwei Li <sup>1</sup>

<sup>1</sup> School of Instrument and Electronics, North University of China, Taiyuan 030051, China

<sup>2</sup> School of Computer Science and Technology, North University of China, Taiyuan 030051, China

\* Correspondence: zhangpeng6@nuc.edu.cn

**Abstract:** Face recognition has achieved great success due to the development of deep convolutional neural networks (DCNNs) and loss functions based on margin. However, complex DCNNs bring a large number of parameters as well as computational effort, which pose a significant challenge to resource-constrained embedded devices. Meanwhile, the popular margin-based loss functions all introduce only one type of margin and cannot further introduce a larger margin to achieve tighter classification boundary. In contrast to the common approach, we believe that additive and multiplicative margins should be used jointly to introduce larger margins from the margin perspective. Therefore, we propose a new margin-based loss function called UnifiedFace. First, we introduce an additive margin in the target angle activation function. Second, we add a multiplicative margin in the non-target angle. UnifiedFace introduces both additive and multiplicative margins, allowing for the introduction of large margins to achieve more compact intra-class variance and closer separated inter-class variance. In addition, we specifically design efficient face recognition models called GhostFaceNet for resource-constrained embedded devices. Experimental results demonstrate that UnifiedFace achieves state-of-the-art performance or performance competed with popular methods in training datasets of different sizes. UnifiedFace achieves optimal performance in models of varying complexity. Moreover, competitive results are achieved in the large-scale test set IJBB/C, especially the state-of-the-art performance achieved in TAR ( $FAR = 1e - 6$ ). GhostFaceNet can significantly improve operational efficiency without significantly degrading recognition performance, making it ideal for embedded devices with limited resources.

**Keywords:** addition margin; multiplicative margin; loss function; face recognition; embedded devices



**Citation:** Zhao, F.; Zhang, P.; Zhang, R.; Li, M. UnifiedFace: A Uniform Margin Loss Function for Face Recognition. *Appl. Sci.* **2023**, *13*, 2350. <https://doi.org/10.3390/app13042350>

Academic Editor: João M. F. Rodrigues

Received: 20 December 2022

Revised: 17 January 2023

Accepted: 20 January 2023

Published: 11 February 2023



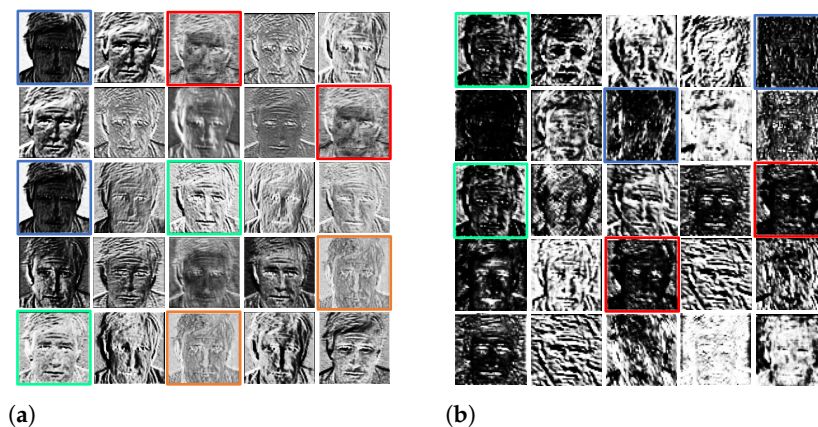
**Copyright:** © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

In recent years, face recognition has achieved remarkable success due to the rapid development of deep convolutional neural networks (DCNNs) [1–3] and loss functions, especially the loss function based on margin [4–9]. Moreover, face recognition technology has been widely used in the fields of finance, public security, 3D sensors in face recognition applications [10], and Crowd counting models adapted to drone-assisted systems [11]. However, complex DCNNs generate a large number of parameters and computational consumption, which is not friendly to resource-limited embedded or mobile devices. Meanwhile, these loss functions embed the face features into the hypersphere space and further increase the intra-class compactness and inter-class separation by introducing margins. Nevertheless, all these margin-based loss functions introduce only one type of margin and cannot further introduce larger margins to achieve tighter classification bounds.

To alleviate the problem of the excessive number of parameters and computational effort associated with complex DCCNs, a series of methods have been proposed to build efficient and lightweight convolutional neural networks, such as knowledge distillation [12,13], networks pruning [14,15], and quantization [16]. In addition, recent research has focused on building efficient convolutional networks with fewer parameters and computational

effort and has achieved great success. Several works [17–19] propose the construction of efficient modules to design lightweight convolutional neural networks. Depthwise separable convolution operations and inverse residual structures are used to construct efficient modules in MobileNet [17,20,21] and obtain competitive performance. To achieve state-of-the-art performance, channel shuffle operation is utilized in ShuffleNet [18]. Although these methods achieve lightweight convolutional neural networks with low computational cost and a low number of parameters by designing efficient units, they do not take redundant information in features into account. For example, as shown in Figure 1a,b, we mark similar features with the same color. GhostNet [19] employs ordinary convolution to generate intrinsic features, followed by inexpensive linear transformations to produce ghost features. The ghost feature reveals sufficient information about the intrinsic features and is able to reduce convolutional consumption with similar recognition performance. Nevertheless, the direct use of the ghost module in face recognition can lead to a significant degradation in model performance. We, therefore, reconstructed an efficient model called GhostFaceNet, based on the ghost module, which significantly reduces the number of parameters and the amount of computation without noticeably degrading performance.



**Figure 1.** Visualization of the different layers of VGG-16, where similar features are annotated with the boxes of the same color: (a) 5th layer of VGG-16 and (b) 10th layer of VGG-16.

Margin-based softmax loss functions are proposed to further reduce intra-class compactness and increase inter-class dispersion. The margin-based loss function usually embeds the face features into the hypersphere space and introduces large margins. The core idea is to redesign the target activation function  $\cos(\theta_y)$  in the softmax cross-entropy loss function.  $\theta_y$  denotes the angle between the sample  $x$  and the weight  $W_y$ , where  $y$  is the true label of the sample  $x$ . The redesigned target activation function needs to be guaranteed to be monotonically decreasing in  $[0, \pi]$ , so that the angle  $\theta_y$  can become smaller when  $\cos(\theta_y)$  reaches the same value. The large margins include additive and multiplicative margins. SphereFace [5] and SphereFaceR [7] further enhance intra-class compactness and inter-class separation by introducing multiplicative margins. CosFace [6,8] and ArcFace [9] achieve the same goal by introducing additive angle margins. Formally, the multiplicative margin is multiplied directly on angle  $\theta_y$ , and the additive margin is added to angle  $\theta_y$  or acts on  $\cos(\theta_y)$ . Furthermore, both approaches have achieved significant performance. Multiplicative and additive margins share a common goal of improving intra-class compactness and inter-class separability by introducing large margins to make decision boundaries tighter. Intuitively, larger margins and stricter decision bounds result by introducing both multiplicative and additive margins.

To effectively unify the additive and multiplicative margins, we propose a unified margin loss function called UnifiedFace. Specifically, the additive and multiplicative margins jointly influence  $\theta_y$  so that it becomes smaller while reaching the same value as  $\cos(\theta_y)$ . We introduce a multiplicative margin in the non-target angle activation function and an additive margin in the target angle activation function. As far as we know, this is the first time that both additive and multiplicative margins have been introduced in the loss function with this idea. Meanwhile, the additive and multiplicative margins are no longer independent but are related. We refer to this approach as UnifiedFace.

Our contributions are summarized as follows:

(1) We propose a new efficient and lightweight model called GhostFaceNet based on ghost module, which significantly reduces the number of parameters and the amount of computation without noticeably degrading performance.

(2) We propose the frame of a unified margin by introducing both additive and multiplicative margins. This frame can effectively introduce larger margins and achieve tighter decision bounds. Moreover, the additive and multiplicative margins are no longer independent of each other but are related.

(3) In this framework, we design a new margin-based loss function called UnifiedFace. UnifiedFace introduces a multiplicative margin in the non-target angle activation function and an additive margin in the target angle activation function. UnifiedFace achieves more stable training and superior generalization. Meanwhile, we give a clear geometric interpretation of UnifiedFace.

The structure of the paper is as follows. Section 2 briefly reviews the work related to deep face recognition and lightweight networks frameworks. Section 3 describes our proposed approach in detail, including the GhostFaceNet and UnifiedFace. Experimental results and related discussions are given in Section 4 and the conclusion is presented in Section 5.

## 2. Related Works

### 2.1. Lightweight Networks Frameworks

The efficient networks structure is an important guarantee for achieving high-performance face recognition. We mainly focus on lightweight networks structures and lightweight face recognition algorithms proposed in recent years.

MobileNetV1 [20] proposes to replace the ordinary convolution with depthwise separable convolution for the first time, which significantly reduces the computational effort and ensures the performance of the model at the same time. In addition, MobileNetV2 [17] designs the inverse residual structure and linear bottleneck layer to make the model structure more reasonable and efficient. MobileNetV3 [21] utilizes complementary search techniques to combine these modules into more efficient models. Group convolutions and channel fusion operations are used in ShuffleNet [18,22] to communicate information from different channels. Ref. [23] uses basic convolution and pooling operations to build a lightweight face detection model and successfully applies it to the corresponding system. The system also performs very well in hazardous situations such as underwater and in avalanches. GhostNet [19] employs ordinary convolution to generate intrinsic features, followed by inexpensive linear transformations to produce ghost features. The ghost feature reveals sufficient information about the intrinsic features and can reduce convolutional consumption with similar recognition performance.

Although excellent results have been achieved with lightweight models, there are no lightweight convolutional neural networks specifically designed for face recognition, especially for resource-constrained embedded devices. MobileFaceNet [24] and ShuffleFaceNet [25] are designed based on MobileNetV2 and ShuffleNetV2, respectively, and achieve significant accuracy. MobileFaceNet designs a global depthwise convolution layer and replaces global average pooling. The use of efficient convolution of variable groups and equivalent angular distillation loss functions in VarGFaceNet [26] makes the model more discriminative and explanatory. MobiFace [27] builds the model using a residual

bottleneck layer with an extended layer and aims to maximize the information in the output vector. It achieves an excellent performance of 99.73% in the LFW dataset. ELANet [2] adds an efficient attention module to MobileNetV2 and applies a feature fusion strategy to significantly improve the performance of lightweight models across postures and ages at a very small computational cost.

## 2.2. Deep Face Recognition

Significant advances in deep face recognition are due to the rapid development of deep convolutional neural networks [28–30]. Early studies consider open-set face recognition to be essentially a multi-classification problem [28,29]. DeepID2 [31] elaborately designs convolutional neural networks (CNNs) and trains CNNs with softmax loss and contrast loss to achieve excellent performance. FaceNet [32] uses triplet loss for training and learns the mapping of face images to Euclidean space directly. Furthermore, it has achieved state-of-the-art results in large-scale face recognition tasks. Center loss [3] enhances the discriminative ability of deep learning features by penalizing the distance between deep features and their corresponding class centers and substantially improves the performance. Other methods based on contrastive loss and triplet loss, such as [33,34], achieve excellent performance and illustrate the importance of metric learning in open set validation.

Margin-based loss function is the main research direction of deep face recognition in recent years and has been widely popular due to its simplicity and effectiveness. L-softmax [4] introduces large angular margins in deep feature learning, and subsequent margin-based loss functions follow this design idea. SphereFace [5] introduces multiplicative margins to achieve tighter decision boundary and introduces weight normalization. To learn large angular margin face embedding features, [35–37] introduce feature normalization. SphereFace does not fully implement the multiplicative margin intuition within  $[0, \pi]$  and its training process is unstable. Therefore, SphereFaceR [7] proposes a simpler and more efficient approach. SphereFaceR can apply multiplicative margins in any of the ranges  $[0, \pi]$ , and SphereFaceR v2 applies margins in non-target angle activation functions for the first time. CosFace [6,8] and ArcFace [9] achieve impressive results by using additive margins instead of multiplicative margins. Specifically, CosFace adds the cosine margin penalty directly to the target logit, and ArcFace adds an angular margin penalty to supervise the network training. Although margin-based loss functions have achieved great success in the field of face recognition, intuitively, introducing both multiplicative and additive margins is more conducive to network training and obtaining superior performance. Other margin-based loss functions achieved significant performance, such as Circle loss [38], Adacos [39], AdaptiveFace [40], CurricularFace [41], and DiscFace [42].

## 3. Our Method

In this section, we first describe our proposed GhostFaceNet-Bottleneck (GF-Bottleneck). The lightweight network framework called GhostFaceNet built from the GF-Bottleneck module is presented in detail. Secondly, we review popular margin-based loss functions in recent years, including additive and multiplicative margins. Finally, we propose a uniform margin method called UnifiedFace and give its clear geometric interpretation.

### 3.1. GhostFaceNet

Based on ghost module, we propose the GhostFaceNet with the configuration shown in Table 1. GhostFaceNet mainly consists of a stack of GhostFaceNet-Bottleneck (GF-Bottleneck).

**Table 1.** The overall framework of GhostFaceNet.  $n$  represents the number of module rereads, and  $s$  indicates the number of similar features generated by each intrinsic feature.

| Input            | Operator            | $n$ | $s$ | Output |
|------------------|---------------------|-----|-----|--------|
| $112 \times 112$ | Conv $3 \times 3$   | -   | 2   | 64     |
| $56 \times 56$   | GF-Bottleneck       | 1   | 2   | 64     |
| $28 \times 28$   | GF-Bottleneck       | 4   | 1   | 64     |
| $28 \times 28$   | GF-Bottleneck       | 1   | 2   | 128    |
| $14 \times 14$   | GF-Bottleneck       | 5   | 1   | 128    |
| $14 \times 14$   | GF-Bottleneck       | 1   | 2   | 256    |
| $7 \times 7$     | GF-Bottleneck       | 4   | 1   | 256    |
| $7 \times 7$     | Conv $1 \times 1$   | -   | 1   | 1024   |
| $7 \times 7$     | GDC                 | -   | 1   | 1024   |
| $1 \times 1$     | Linear $1 \times 1$ | -   | 1   | 128    |

In the ghost module, the input features are first convolved by ordinary convolution to generate a fixed number  $c'$  of intrinsic features  $Y \in \mathbb{R}^{w' \times h' \times c'}$ :

$$Y = X * f \quad (1)$$

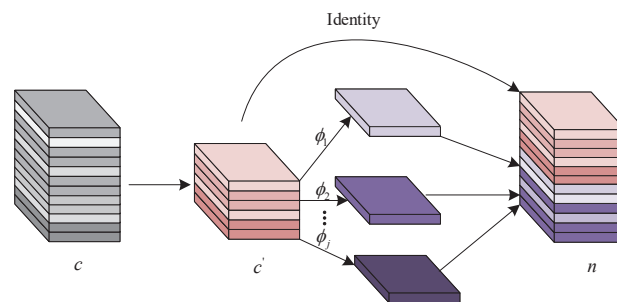
where  $X \in \mathbb{R}^{w \times h \times c}$  is the input feature and  $f \in \mathbb{R}^{k \times k \times c \times c'}$  is the convolution kernel. The bias terms are ignored for simplicity. Then, the intrinsic features generate ghost features  $y_{ij}$ , i.e., features associated with the intrinsic features, by some series of linear operations (LO):

$$y_{ij} = \phi_{ij}(y_i'), i = 1, 2, \dots, m, j = 1, 2, \dots, s \quad (2)$$

where  $y_{ij}$  is the  $i$ -th feature in intrinsic feature  $Y$  and  $\phi_{ij}$  is the linear operation that generates the  $j$ -th associated feature.  $m$  is the number of generated intrinsic features, and  $s$  represents the number of ghost features generated for each intrinsic feature. The mathematical relationship between  $m$  and  $s$  is as follows:

$$n = m \times s \quad (3)$$

where  $n$  is the number of channels of the final output feature. Feature information  $y_{ij}$  and intrinsic features are connected together to output the final feature information. The operation flow of the ghost module is shown in Figure 2.



**Figure 2.** Framework of ghost modules.  $c$ ,  $c'$ , and  $n$  represent the number of input feature channels, the number of intrinsic features, and the number of output features, respectively.

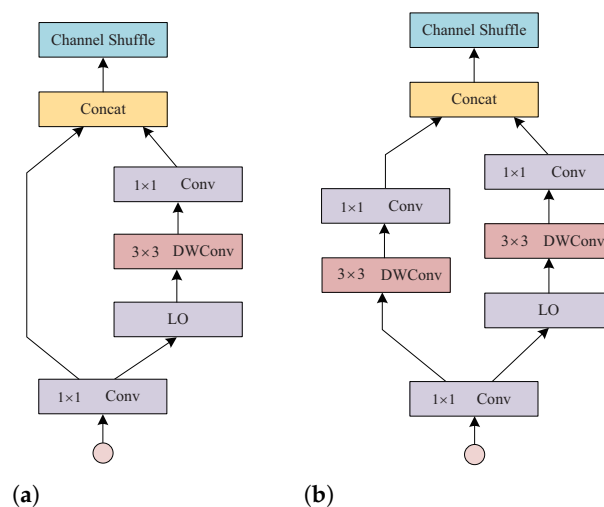
GF-Bottleneck mainly consists of ghost modules, depthwise separable convolutions, and  $1 \times 1$  convolutions, as shown in Figure 3. The input features  $X \in \mathbb{R}^{w \times h \times c}$  are first divided into two branches by normal convolution in DG module. A branch consists of LO in DG modules, a depthwise separable convolution with a step size of 1, and  $1 \times 1$  convolution. The other one acts as an identity. We define the output ratio input as the output expansion ratio as follows:



$$r = \frac{n}{c} \quad (4)$$

According to ShuffleNetV2 [18], we set  $r = 1$  for all modules, to reduce memory access cost(MAC). The feature information of the two branches is connected by the concat operation and finally output by the channel shuffle operation to allow the communication of the different channel features.

We construct a new structure to increase the number of output features as well as spatial downsampling, as illustrated in Figure 3b. Unlike the previous case with stride = 1, in the identity branch, we add the depthwise separable convolution and  $1 \times 1$  convolution to enrich feature expression. In addition, we set stride = 2.



**Figure 3.** Structural design of GF-Bottleneck. Ghost module consists of normal convolution and linear operations (LO). DWConv represents depthwise separable convolutions. (a) Stride = 1 and (b) Stride = 2.

Based on GF-Bottleneck, we propose the GhostFaceNet with the configuration shown in Table 1. GhostFaceNet mainly consists of a stack of GF-Bottleneck.

The first layer of the network is a standard  $3 \times 3$  convolution used for fast downsampling. A series of GF-Bottleneck modules are applied, and the number of feature channels is gradually increased. Global average pooling (GAP) treats each unit of the output features with the same importance. However, it contradicts the theory that different units contribute differently to the face feature vector. Refs. [9,43] show that GAP is inaccurate for face recognition tasks. The global depthwise convolution (GDC) in [20] assigns different weights to units with various degrees of importance. Meanwhile, the fully connected (FC) layer generates more weight information and thus increases the size of the model. In GhostFaceNet, GDC is used instead of GAP. PReLU [44] is chosen to be nonlinear to replace ReLU [45]. Finally, a linear  $1 \times 1$  convolution is used to output compact 128-dimensional face features. GhostFaceNet has 0.82 M parameters and 0.15 GFLOPs, far less than the parameters and computations of other lightweight networks. Meanwhile, GhostFaceNet can well verify the performance of UnifiedFace in lightweight networks.

### 3.2. Margin-Based Loss Functions

The traditional softmax loss function, which does not explicitly encourage enhanced intra-class compactness and inter-class separation, is expressed as follows:

$$L_{softmax} = -\frac{1}{N} \sum_i \log\left(\frac{e^{W_y^T x_i + b_y}}{\sum_j^n e^{W_j^T x_i + b_j}}\right) \quad (5)$$

where  $N$  is batch size and  $n$  is the number of classes.  $b_y$  indicates the bias term for the  $y$ -th class.

To effectively alleviate this problem, margin-based softmax loss functions are proposed with great success. Commonly, the margin-based softmax constraint redefines logit as the following equation:

$$W_j^T x_i = \|W_j\| \|x_i\| \cos\theta_j \quad (6)$$

where  $\theta_j$  denotes the angle between the feature and the  $j$ -th weight vector. The feature is normalized by  $L_2$  normalization and then re-specified as  $s$ . The weight is fixed to 1 by  $L_2$  normalization.

In [7],  $\cos\theta$  is replaced by the target angle activation function (non-target angle activation function). The margin-based softmax is uniformly defined as follows

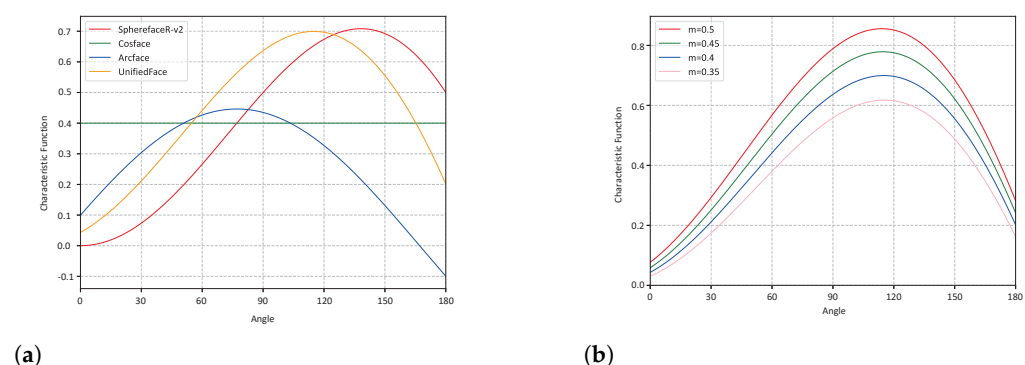
$$L = \log(1 + \sum_{i \neq y} e^{Q(\theta_y, \theta_i, s, m)}) \quad (7)$$

$$Q(\theta_y, \theta_i, s, m) = s \cdot (\eta(\theta_i) - \eta(\theta_y) + \Delta(\theta_y)) \quad (8)$$

The  $m$  indicates the margin, including additive margin or multiplicative margin.  $\eta(\theta_i) - \eta(\theta_y)$  denotes the confidence difference of various classes.  $\Delta(\theta)$  is the characteristic function to describe angular margins, including the size of the margin and the training stability. The characteristic function is defined as follows:

$$\Delta(\theta) = \eta(\theta) - \psi(\theta) > 0 \quad (9)$$

where  $\eta(\theta)$  and  $\psi(\theta)$  are defined as the target activation function and non-target activation function, respectively. In particular, different  $\eta(\theta)$  and  $\psi(\theta)$  constitute various margin-based loss functions, as shown in Table 2. A comparison of the different margin-based loss functions for the characteristic functions is given in Figure 4a. Despite the great progress of these margin-based loss functions, the problem of how to introduce margins more efficiently remains a tricky one. That is, designing a simple and excellent performance angular margin is still a problem. It is natural to think of unifying additive and multiplicative margins to solve this problem.



**Figure 4.** (a) Comparison between UnifiedFace and existing methods (b) Comparison of UnifiedFace at different  $m$ .

**Table 2.** Different margin-based loss functions for the characteristic functions.

| Method         | Non-Target Function        | Target Function                                  | Characteristic Function   |
|----------------|----------------------------|--|---|
| SphereFace     | $\cos(\theta)$             | $(-1)^k \cos(m\theta) - 2k$                      | $\cos(\theta) - (-1)^k \cos(m\theta) + 2k$                      |
| SphereFaceR v1 | $\cos(\theta)$             | $\cos(\min(m, \frac{\pi}{\theta}) \cdot \theta)$ | $\cos(\theta) - \cos(\min(m, \frac{\pi}{\theta}) \cdot \theta)$ |
| SphereFaceR v2 | $\cos(\frac{\theta}{m})$   | $\cos(\theta)$                                   | $\cos(\frac{\theta}{m}) - \cos(\theta)$                         |
| CosFace        | $\cos(\theta)$             | $\cos(\theta) - m$                               | $m$   |
| ArcFace        | $\cos(\theta)$             | $\cos(\theta + m)$                               | $\cos(\theta) - \cos(\theta + m)$                               |
| UnifiedFace    | $\cos(\frac{\theta+m}{M})$ | $\cos(\theta + m)$                               | $\cos(\frac{\theta+m}{M}) - \cos(\theta + m)$                   |

### 3.3. UnifiedFace

To unify the additive and multiplicative margins, an additive margin is introduced in the target angle activation function  $\psi(\theta)$  and a multiplicative margin is added in the non-target angle activation function  $\eta(\theta)$ . Following the conventional additive margin loss function [9], we first designed  $\psi(\theta)$ , as follows:

$$\psi(\theta) = \cos(\theta + m) \quad (10)$$

where  $m$  is usually an additive margin penalty.

In addition, to design a reasonable loss function with multiplicative margin, we construct  $\psi(\theta)$  and  $\eta(\theta)$  satisfying the following relation:

$$\psi(\theta) = \eta(M\theta), \theta \in [0, \pi] \quad (11)$$

where  $M$  is usually the specified normal number. Therefore, we design a novel non-target activation function  $\eta(\theta)$ :

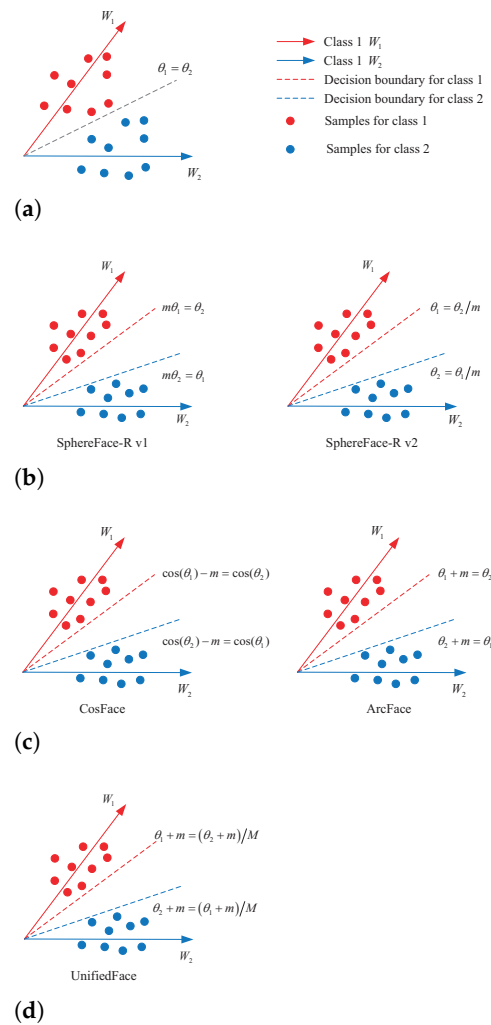
$$\eta(\theta) = \cos(\frac{\theta + m}{M}) \quad (12)$$

where  $M = e^m$ .

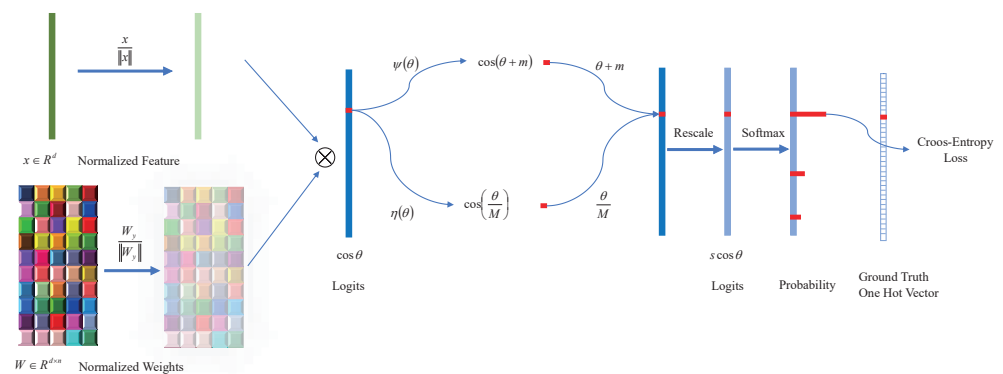
From Figure 4b, we can observe the change in the trend of UnifiedFace at different  $m$  in detail. The characteristic functions at different values of  $m$  all follow the angular margin minimum for simple samples and maximum for medium hard samples. Meanwhile, a sufficiently large angular margin is given for hard samples. Compared with other popular loss functions of characteristic functions, UnifiedFace can obtain the maximum angular margin for medium hardness samples. Furthermore, UnifiedFace does not give too much angular margin to hard samples leading to an excessive focus on these hard samples during training compare to SphereFaceR v2.

To facilitate understanding, the binary case is considered to illustrate intuitively the uniform additive and multiplicative margins. Figure 5 intuitively compares the no angular margin, multiplicative angular margin, additive angular margin, and uniform angular margin. For a sample  $x$ , if no margin is introduced,  $\cos(\theta_1) = \cos(\theta_2)$  becomes the decision boundary for class 1, i.e., the decision boundary is  $\theta_1 = \theta_2$ , as illustrated in Figure 5a. When both additive and multiplicative margins are introduced, it also means that  $\psi(\theta) = \psi_1(\theta)$  and  $\eta(\theta) = \eta_1(\theta)$ .  $\cos(\theta_1 + m) = \cos(\frac{\theta_2 + m}{M})$  is the decision boundary for class 1. When the sample  $x$  belongs to class 1, it is required that  $\cos(\theta_1 + m) > \cos(\frac{\theta_2 + m}{M})$ . Correctly classifying sample  $x$  into class 1 requires  $\theta_1 + m < \frac{\theta_2 + m}{M}$ , while correctly classifying sample  $x$  into class 2 requires  $\theta_2 + m < \frac{\theta_1 + m}{M}$ . As shown in Table 3, the decision boundary of UnifiedFace is more stringent. That is when the sample  $x$  belonging to class 1 is correctly classified,  $\theta_1$  is smaller than the other loss functions of both additive margin and multiplicative margin. The process of UnifiedFace supervise face recognition network is illustrated in Figure 6.





**Figure 5.** (a) No angular margin; (b) is the multiplicative margin; (c) is the additive margin; (d) is the uniform angular margin.



**Figure 6.** Detailed supervision of DCNN by UnifiedFace loss function. After normalizing the features  $x$  and weights  $W_y$ , we obtain the logit vector  $\cos \theta$ . After that, the angle  $\theta$  between the weights and the features is obtained. Unlike previous methods, the obtained target angle pinch angle is given an additive margin, i.e.,  $\theta + m$ , and the obtained non-target angle is given a multiplicative margin, i.e.,  $\frac{\theta}{M}$ . Our approach leads to features learning more compact intra-class similarity and inter-class separability. Then, all logits are multiplied by the feature scale  $s$ . Finally, the logit contributes to the cross-entropy loss through the softmax function.

**Table 3.** Comparison of binary bounds for different loss functions.

| Method      | Binary Boundary for Class 1             |
|-------------|---|
| softmax     | $\theta_1 < \theta_2$                   |
| SphereFace  | $m\theta_1 < \theta_2$                  |
| CosFace     | $\cos(\theta_1) - m < \theta_2$         |
| ArcFace     | $\theta_1 + m < \theta_2$               |
| UnifiedFace | $\theta_1 + m < \frac{\theta_2 + m}{M}$ |

#### 4. Experiments

In this section, we first introduce the dataset used for the experiment, the network structure, and the detailed setup of the experiment. In addition, we give a detailed description of the network structure. Secondly, we conducted ablation studies on different training datasets separately to investigate the effect of their hyperparameters on UnifiedFace. Then, we give the results of comparing UnifiedFace with other state-of-the-art methods. Meanwhile, we report the performance comparison results of UnifiedFace with popular loss functions in networks of different complexity. Finally, we evaluate and compare with state-of-the-art methods in the popular large-scale test IJBB/C.

##### 4.1. Datasets

(1) Training datasets: We select VGGFace2 [46] and MS1M V3 as the training dataset. To illustrate the generalization performance of our proposed method, we selected the large-scale dataset MS1M V3 and the small-scale dataset VGGFace2 for model training, respectively. VGGFace2 includes 3.14M images of faces in different poses, ages, and ethnicities. The 5.1 million face images of 93K identities are included in MS1MV3, which is a cleaned Ms-celeb-1m dataset [47]. No data enhancement is used in the training process.

(2) Validation and test datasets: We choose LFW [48], CPLFW [49], CALFW [50], CFP [51], VGGFace2-FP [46], IJB-B [52], and IJB-C [53] datasets to validate and evaluate our method. Table 4 shows the statistics of the datasets.

**Table 4.** Statistical information on the datasets used.

| Dataset         | ID    | Images  |
|-----------------|-------|---------|
| MS1M-RetinaFace | 93 K  | 5.1 M   |
| VGGFace2        | 8.6 K | 3.1 M   |
| LFW             | 5749  | 13,233  |
| CPLFW           | 5749  | 12,174  |
| CALFW           | 5749  | 11,652  |
| CFP             | 500   | 7000    |
| VGGFace2-FP     | 500   | 173 K   |
| IJBB            | 1845  | 76.8 K  |
| IJBC            | 3531  | 148.8 K |

##### 4.2. Implementation

We use PyTorch [54] to implement all the experiments. We use GhostFaceNet as the backbone network to investigate the performance of the proposed new efficient lightweight model in using UnifiedFace. In addition, we use ResNet-100 [30] as the backbone network. The outputs of ResNet-100 are all 512 dimensions for a fair comparison. The batch size is set to 256. A value of 0.1 is used as the initial learning rate and divided by 10 every 10 epochs. The training ends at 25 epochs. The momentum and the weight decay are set to 0.9 and  $5e - 5$ , respectively. In all experiments, we compare the accuracy of the loss function in different datasets.

### 4.3. Ablation Experiments

#### 4.3.1. Effect of Hyperparameters $M$ Furthermore, $S$

We use VGGFace2 and MS1M V3 as training datasets to explore the effect of hyperparameters on the performance of UnifiedFace in training datasets of different scales, respectively.

We first investigate the advantages and disadvantages of the performance of different hyperparameters in the small-scale training dataset VGGFace2, and the experimental results are shown in Table 5. The hyperparameter  $s$  is set to 64 and 40, and the hyperparameter  $m$  is fixed from 0.35 to 0.5. Experimental results show that UnifiedFace performs well for values of  $m$  ranging from 0.35 to 0.4 in small training datasets. In particular, when  $s = 64$  and  $m = 0.4$ , UnifiedFace achieves the optimal performance. Specifically, the optimal setting for the additive margin of UnifiedFace is between 0.35 and 0.4. The optimal setting for the multiplicative margin of UnifiedFace is between 1.4 and 1.5.

Then, we report the effect of hyperparameters on performance in the large-scale training dataset MS1M V3, and the experimental results are presented in Table 6. The hyperparameters  $s$  are set to 64 and 40, and the hyperparameters  $m$  are fixed from 0.35 to 0.5. In particular, the model performance is optimal when  $s = 50$ ,  $m = 0.4$ . The experimental results show that the optimal setting interval for the hyperparameter  $m$  is from 0.4 to 0.45, which means that the optimal setting range for the additive margin is from 0.4 to 0.45, and the optimal setting range for the multiplicative margin is from 1.5 to 1.6. The results of hyperparametric experiments on training datasets of different sizes show that UnifiedFace has excellent generalization ability.

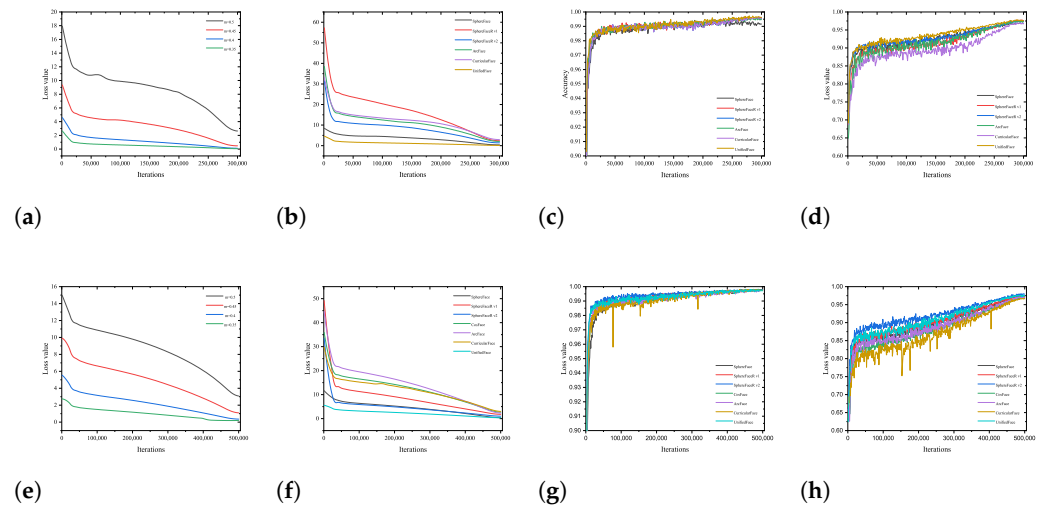
In addition, we give the variation of loss values during UnifiedFace training with different hyperparameters, as shown in Figure 7a,e. During the training process, different hyperparameters lead to different values of the final convergence of the loss values.  $s = 64$ ,  $m = 0.5$ , the loss values fluctuate the most. In other cases, the loss value starts with a sharp drop and then falls steadily. Although the loss value fluctuation during the training process is smoothest and converges to a minimum value when  $m$  is taken as small as possible, the performance is not the best. We select the best-performing hyperparameters as the experimental parameters and compare the change of loss values during the training process with the popular loss functions in recent years, as illustrated in Figure 7b,f.

**Table 5.** The ablation study on different hyperparameters  $s$  and  $m$  with ResNet-100. The VGGFace2 dataset, which is relatively small in size, is chosen for the training dataset.

| $s$ | $m$  | LFW    | CALFW  | CPLFW  | VGG2_FP | CFP    |        |
|-----|------|--------|--------|--------|---------|--------|--------|
|     |      |        |        |        |         | CFP_FF | CFP_FP |
| 64  | 0.5  | 99.55% | 93.08% | 91.92% | 95.14%  | 99.34% | 97.47% |
| 64  | 0.45 | 99.52% | 92.80% | 92.23% | 94.80%  | 99.26% | 97.01% |
| 64  | 0.4  | 99.73% | 93.77% | 92.58% | 95.36%  | 99.37% | 97.83% |
| 40  | 0.35 | 99.67% | 93.10% | 92.48% | 95.76%  | 99.26% | 97.59% |

**Table 6.** The ablation study on different hyperparameters  $s$  and  $m$  with ResNet-100. The large-scale MS1M V3 dataset is chosen as the training dataset.

| $s$ | $m$  | LFW    | CALFW  | CPLFW  | VGG2_FP | CFP    |        |
|-----|------|--------|--------|--------|---------|--------|--------|
|     |      |        |        |        |         | CFP_FF | CFP_FP |
| 64  | 0.5  | 99.77% | 96.00% | 92.42% | 95.00%  | 99.77% | 97.09% |
| 64  | 0.45 | 99.80% | 96.00% | 92.68% | 95.34%  | 99.77% | 97.30% |
| 50  | 0.4  | 99.80% | 96.17% | 92.65% | 96.52%  | 99.79% | 97.77% |
| 40  | 0.35 | 99.78% | 95.90% | 92.53% | 96.02%  | 99.76% | 97.83% |



**Figure 7.** (a,e) denote the change in loss values of UnifiedFace using different training datasets with various hyperparameters, respectively. (b,f) indicate the comparison of loss values of UnifiedFace using different datasets with state-of-the-art methods. (c,g) represent the comparison of UnifiedFace using various datasets with state-of-the-art methods in the LFW validation dataset. (d,h) show the comparison of UnifiedFace using different datasets compared with popular methods in the CFPFP validation dataset.

#### 4.3.2. The Comparison with State-of-the-Art Methods

The performance comparisons of UnifiedFace with other state-of-the-art methods at different scales of training datasets are given in Tables 7 and 8, respectively. UnifiedFace outperforms other state-of-the-art methods when using small-scale datasets as training datasets. Specifically, UnifiedFace achieves state-of-the-art performance in the validation datasets LFW, CPLFW, and CFP\_FP. The performance of UnifiedFace improves over the optimal loss function by 0.08%, 0.08%, and 0.04% in these three validation datasets, respectively. Competitive performance is also achieved in the remaining three validation datasets. UnifiedFace also outperforms the current popular loss function when using MS1M V3 as the training dataset. Specifically, UnifiedFace achieves state-of-the-art performance in CALFW, VGG2\_FP, and CFP\_FF. UnifiedFace substantially improves the performance of the model in the VGG2\_FP validation dataset. Furthermore, UnifiedFace achieves suboptimal performance in all other validation sets. A comparison of UnifiedFace with popular loss functions in recent years in training sets of different sizes illustrates its excellent generalization ability.

**Table 7.** Performance comparison of UnifiedFace with state-of-the-art methods in using VGGFace2 as the training dataset.

| Method         | s  | m    | LFW    | CALFW  | CPLFW  | VGG2_FP | CFP    |        |
|----------------|----|------|--------|--------|--------|---------|--------|--------|
|                |    |      |        |        |        |         | CFP_FF | CFP_FP |
| SphereFace     | -  | 4    | 99.43% | 91.18% | 90.33% | 95.46%  | 98.90% | 96.99% |
| SphereFaceR v1 | 64 | 1.5  | 99.63% | 93.70% | 92.50% | 95.08%  | 99.46% | 97.79% |
| SphereFaceR v2 | 64 | 1.5  | 99.65% | 93.37% | 92.25% | 95.30%  | 99.37% | 97.50% |
| CosFace        | 64 | 0.45 | 99.63% | 93.83% | 92.37% | 94.40%  | 99.37% | 97.10% |
| ArcFace        | 64 | 0.45 | 99.62% | 93.77% | 92.33% | 94.90%  | 99.40% | 97.53% |
| CurricularFace | 64 | 0.5  | 99.62% | 93.90% | 91.58% | 94.46%  | 99.43% | 96.90% |
| UnifiedFace    | 64 | 0.4  | 99.73% | 93.77% | 92.58% | 95.36%  | 99.37% | 97.83% |

**Table 8.** Performance comparison of UnifiedFace with state-of-the-art methods in using MS1M V3 as the training dataset.

| Method         | s  | m    | LFW    | CALFW  | CPLFW  | VGG2_FP | CFP    |        |
|----------------|----|------|--------|--------|--------|---------|--------|--------|
|                |    |      |        |        |        |         | CFP_FF | CFP_FP |
| SphereFace     | -  | 3    | 99.80% | 96.00% | 92.42% | 95.34%  | 99.73% | 97.57% |
| SphereFaceR v1 | 64 | 1.5  | 99.78% | 96.03% | 92.55% | 95.04%  | 99.78% | 97.39% |
| SphereFaceR v2 | 64 | 1.5  | 99.82% | 96.03% | 92.97% | 95.69%  | 99.77% | 98.00% |
| CosFace        | 64 | 0.45 | 99.82% | 96.08% | 92.27% | 94.38%  | 99.76% | 96.89% |
| ArcFace        | 64 | 0.45 | 99.77% | 96.12% | 92.43% | 94.66%  | 99.76% | 97.13% |
| CurricularFace | 64 | 0.5  | 99.80% | 96.02% | 92.05% | 94.06%  | 99.77% | 96.77% |
| UnifiedFace    | 50 | 0.4  | 99.80% | 96.17% | 92.65% | 96.52%  | 99.79% | 97.77% |

In addition, the comparison of UnifiedFace with the popular loss function in recent years in terms of loss value change during training is given. Both Figure 7b,f show that UnifiedFace has the most stable loss value change process and the smallest convergence value during the training process. Meanwhile, there is no large and drastic loss change during the training process. The accuracy comparison of UnifiedFace with popular loss functions in LFW and CFP\_FP validation datasets during training is shown in Figure 7c,h, respectively.

#### 4.4. Experiments in GhostFaceNet

We report on the performance of GhostFaceNet trained using UnifiedFace., as shown in Table 9. Simultaneously, the performance comparison of UnifiedFace with popular loss functions in lightweight networks is given.

The popular loss functions of recent years are used to train GhostFaceNet and compare it with UnifiedFace. The margins of SphereFace, CosFace, and AirFace [55] are set to 4, 0.3, and 0.45, respectively. AirFace is a loss function specifically designed for lightweight network models, and its linear target logit makes the training process very stable. The margins of ArcFace are given as 0.15, 0.30, and 0.35, and are pre-trained by softmax before using ArcFace. On LFW and CFP\_FP, UnifiedFace is better than the other loss functions. In particular, UnifiedFace substantially improves the performance of the model in the cross-pose verification set CFP\_FP. Furthermore, in CALFW, UnifiedFace achieves similarly competitive results. The experimental results also demonstrate the degradation of model performance caused by excessive margins in lightweight networks when using margin-based loss functions for supervised training. For example, the performance for the case of  $m = 0.45$  is lower than that of  $m = 0.35$  in UnifiedFace. Likewise, the performance in ArcFace is similar. The fact that UnifiedFace still achieves optimal performance in lightweight networks despite the introduction of large margins illustrates the advantage of introducing both additive and multiplicative margins in UnifiedFace.

**Table 9.** Performance comparison between UnifiedFace and popular loss functions in lightweight networks.

| Loss        | m    | LFW    | CALFW  | CFP_FP |
|-------------|------|--------|--------|--------|
| SphereFace  | 4    | 99.47% | 94.97% | 91.79% |
| CosFace     | 0.3  | 99.35% | 94.27% | 92.23% |
| ArcFace     | 0.15 | 99.53% | 94.58% | 94.21% |
| ArcFace     | 0.3  | 99.52% | 94.63% | 93.46% |
| ArcFace     | 0.35 | 99.50% | 94.67% | 93.10% |
| AirFace     | 0.45 | 99.48% | 94.98% | 92.73% |
| UnifiedFace | 0.45 | 99.60% | 94.53% | 92.74% |
| UnifiedFace | 0.4  | 99.63% | 94.80% | 93.70% |
| UnifiedFace | 0.35 | 99.50% | 94.63% | 94.54% |

#### 4.5. Comparison Experiments with Different Lightweight Network Models

We compared the performance of GhostFaceNet with popular models in recent years, including the complex model ResNet-50 and other popular lightweight models. We used UnifiedFace as the loss function.

The performance of GhostFaceNet is shown in Table 10. Although GhostFaceNet achieves the best performance in the validation dataset LFW, it does not perform well in the cross-pose and cross-age validation datasets. GhostFaceNet does not have a large enough number of channels to learn richer features. Nevertheless, GhostFaceNet generates the lowest number of parameters and computation, especially since it consumes only 0.15 G of computation. Although GhostFaceNet does not perform well in the cross-pose and cross-age validation datasets, its overall performance is still competitive. Meanwhile, it is very computationally small and can significantly reduce the inference time of the model in resource-constrained devices.

**Table 10.** Performance comparison of GhostFaceNet with other models. Results are in % and higher number indicates better performance.

| Model         | m    | LFW    | CALFW  | CFP-FP | PARAM   | FLOPs  |
|---------------|------|--------|--------|--------|---------|--------|
| ResNet-50     | 0.45 | 99.61% | 94.81% | 95.17% | 40.25 M | 2.19 G |
| EfficientNet  | 0.45 | 99.52% | 95.31% | 96.55% | 6.58 M  | 1.14 G |
| MobileNetV2   | 0.45 | 99.55% | 94.87% | 93.40% | 2.26 M  | 0.43 G |
| MobileFaceNet | 0.45 | 99.53% | 94.95% | 95.49% | 1.0 M   | 0.45 G |
| GhostNet      | 0.45 | 99.58% | 94.78% | 95.40% | 4.06 M  | 0.44 G |
| GhostFaceNet  | 0.45 | 99.60% | 94.53% | 92.74% | 0.82 M  | 0.15 G |
| GhostFaceNet  | 0.4  | 99.63% | 94.80% | 93.70% | 0.82 M  | 0.15 G |
| GhostFaceNet  | 0.35 | 99.50% | 94.63% | 94.54% | 0.82 M  | 0.15 G |

#### 4.6. Experiments Evaluated on IJBB/C

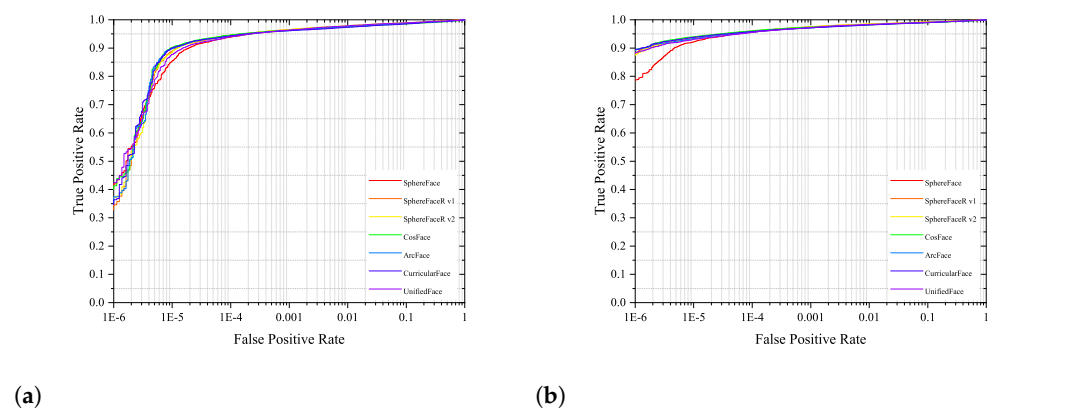
In this section, we evaluate the performance of UnifiedFace on popular large-scale benchmarks (i.e., IJB-B and IJB-C). A fair comparison with state-of-the-art methods is provided. Specifically, we train UnifiedFace on MS1M V3. We compare UnifiedFace with the current state-of-the-art approaches in recent years, i.e., SphereFace [5], SphereFaceR [7], CosFace [6,8], ArcFace [9], and CurricularFace [41].

The experimental results in Table 11 show that the experimental results in the large-scale test datasets are consistent with those in the validation datasets. This consistency also indicates the validity of the selected test datasets and models. It is also well demonstrated that the models that obtained good performance in the validation datasets can also achieve correspondingly good results in the test datasets. From Table 11, we can observe that although different types of margins produce similar performance, result in different results at a low false acceptance rate (FAR). UnifiedFace can significantly improve the performance on the IJBB and IJBC test datasets. In particular, at TAR (@FAR =  $1e-6$ ), UnifiedFace achieves state-of-the-art performance. Moreover, UnifiedFace performs particularly well at other low FAR, which is of great interest for designing robust face recognition. The receiver operating characteristic curves(ROC) of UnifiedFace with other loss functions are given in Figure 8a,b, respectively.



**Table 11.** Evaluation on IJB-B and IJB-C. We use ResNet-100 as the backbone architecture and MS1MV3 as the training dataset for all the compared methods. Results are in % and higher number indicates better performance.

| Method               | m    | IJB-B             |                   |                   |                   | IJB-C             |                   |
|----------------------|------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|
|                      |      | $1 \times e^{-6}$ | $1 \times e^{-5}$ | $1 \times e^{-4}$ | $1 \times e^{-6}$ | $1 \times e^{-5}$ | $1 \times e^{-4}$ |
| SphereFace           | 3    | 42.00             | 85.84             | 93.90             | 78.77             | 92.18             | 95.68             |
| SphereFaceR v1       | 1.5  | 34.56             | 88.88             | 94.47             | 88.39             | 93.69             | 95.97             |
| SphereFaceR v2       | 1.5  | 37.22             | 88.45             | 94.51             | 87.79             | 93.34             | 95.99             |
| CosFace              | 0.45 | 41.29             | 90.21             | 94.63             | 89.61             | 94.02             | 96.11             |
| ArcFace              | 0.45 | 37.41             | 89.82             | 94.20             | 89.42             | 93.38             | 95.85             |
| CurricularFace       | 0.45 | 36.38             | 90.06             | 94.37             | 89.58             | 93.83             | 95.85             |
| UnifiedFace(MS1M V3) | 0.4  | 42.51             | 88.52             | 94.03             | 88.50             | 93.12             | 95.62             |



**Figure 8.** Comparison of the ROC curves of UnifiedFace and the popular loss function. (a) ROC for IJB-B; (b) ROC for IJB-C.

#### 4.7. Experiments in Embedded Devices

We conducted a 1:1 speed test on a computer platform (i9-12900K CPU). In addition, we also performed accuracy test experiments on the computer platform with the LFW dataset, as illustrated in Table 12. GhostFaceNet achieve the best performer in the LFW and is 15.3ms faster than EfficientNet, MobileNetV2, and MobileFaceNet, 2.39 times, 1.67 times, and 1.55 times, respectively.

Meanwhile, we conduct running speed comparison experiments in resource-constrained embedded device to further illustrate the advantages of GhostFaceNet in embedded devices. The embedded platform of choice is the ZYNQ 7020, which uses ARM+FPGA SOC technology to integrate ARM Cortex-A9 and FPGA programmable logic on a single chip. We use MTCNN [56] for face images detection and crop it to  $112 \times 112$ . We use NCNN for PyTorch model transformation and implement the final run-time test. We do not apply quantization to the model, so there is no degradation in the model recognition capability.

**Table 12.** Lightweight model performance testing in computer platform.

| Method        | Accuracy | Speed    |
|---------------|----------|----------|
| ResNet-50     | 99.64%   | -        |
| EfficientNet  | 99.53%   | 36.5 ms  |
| MobileNetV2   | 99.55%   | 25.67 ms |
| MobileFaceNet | 99.53%   | 23.67 ms |
| GhostFaceNet  | 99.63%   | 15.3 ms  |

The runtime results of GhostFaceNet and other methods for single-face image recognition in embedded devices are shown in Table 13. GhostFaceNet is 11.04 times, 8.57 times, 2.75 times, and 2.82 times faster than ResNet50, EfficientNet, MobileNetV2, and MobileFaceNet, respectively. In general, GhostFaceNet outperforms other lightweight models in embedded devices in terms of speed and is suitable for deployment in resource-limited embedded devices. For example, in practical scenarios where system resources are limited, such as time and attendance card punching, community access control, and surveillance devices, it becomes very difficult to deploy complex models, and our proposed GhostFaceNet can perform the relevant recognition tasks very easily.

**Table 13.** Speed comparison of GhostFaceNet with different models for single image recognition in embedded devices.

| Method        | Embedded Platform | PARAM   | FLOPs  |
|---------------|-------------------|---------|--------|
| ResNet-50     | 3.09 s            | 40.29 M | 2.19 G |
| EfficientNet  | 2.4 s             | 6.58 M  | 1.14 G |
| MobileNetV2   | 0.77 s            | 2.26 M  | 0.43 G |
| MobileFaceNet | 0.79 s            | 1.0 M   | 0.45 G |
| GhostFaceNet  | 0.28 s            | 0.82 M  | 0.15 G |

## 5. Conclusions

We propose an efficient lightweight convolutional neural network for embedded or mobile devices, called GhostFaceNet, that drastically reduces the amount of computation required by the model. In addition, our paper proposes a unified framework. In this framework, the intuition of introducing larger margins is effectively achieved by introducing both additive and multiplicative margins. We have conducted extensive experiments on many validation datasets and popular benchmark tests to verify the effectiveness of our proposed algorithm. Furthermore, during the actual training process and system testing, we found that our proposed loss function can introduce larger margins without causing training instability, as well as GhostFaceNet is able to effectively reduce the inference time of the model in edge devices. In future work, we will also design more efficient lightweight modules from the perspective of hardware system resources and take advantage of the highly parallel nature of FPGAs to further accelerate the inference speed of the models in edge devices. Moreover, we continue to explore the design of more efficient margins.

**Author Contributions:** Conceptualization, F.Z.; methodology, F.Z.; software, F.Z.; investigation, P.Z.; writing—original draft preparation, F.Z.; writing—review and editing, P.Z.; supervision, R.Z.; project administration, M.L.; funding acquisition, P.Z. All authors have read and agreed to the published version of the manuscript.

**Funding:** This publication was funded by the Technology Area Fund under Grant No. 2021-JCJQ-JJ-0726.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** We certify that there are no actual or potential conflict of interest in relation to this article. The authors declare that they have no competing interests. We certify that there are no financial interests in relation to this article.

## References

1. Wang, Q.; Guo, G. LS-CNN: Characterizing local patches at multiple scales for face recognition. *IEEE Trans. Inf. Forensics Secur.* **2019**, *15*, 1640–1653. [[CrossRef](#)]
2. Zhang, P.; Zhao, F.; Liu, P.; Li, M. Efficient Lightweight Attention Network for Face Recognition. *IEEE Access* **2022**, *10*, 31740–31750. [[CrossRef](#)]
3. Wen, Y.; Zhang, K.; Li, Z.; Qiao, Y. A discriminative feature learning approach for deep face recognition. In *European Conference on Computer Vision*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 499–515.

4. Liu, W.; Wen, Y.; Yu, Z.; Yang, M. Large-margin softmax loss for convolutional neural networks. *arXiv* **2016**, arXiv:1612.02295.
5. Liu, W.; Wen, Y.; Yu, Z.; Li, M.; Raj, B.; Song, L. Sphereface: Deep hypersphere embedding for face recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 212–220.
6. Wang, H.; Wang, Y.; Zhou, Z.; Ji, X.; Gong, D.; Zhou, J.; Li, Z.; Liu, W. Cosface: Large margin cosine loss for deep face recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 5265–5274.
7. Liu, W.; Wen, Y.; Raj, B.; Singh, R.; Weller, A. SphereFace Revived: Unifying Hyperspherical Face Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *45*, 2458–2474. [[CrossRef](#)] [[PubMed](#)]
8. Wang, F.; Cheng, J.; Liu, W.; Liu, H. Additive margin softmax for face verification. *IEEE Signal Process. Lett.* **2018**, *25*, 926–930. [[CrossRef](#)]
9. Deng, J.; Guo, J.; Xue, N.; Zafeiriou, S. Arcface: Additive angular margin loss for deep face recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 4690–4699.
10. Ulrich, L.; Vezzetti, E.; Moos, S.; Marcolin, F. Analysis of RGB-D camera technologies for supporting different facial usage scenarios. *Multimed. Tools Appl.* **2020**, *79*, 29375–29398. [[CrossRef](#)]
11. Woźniak, M.; Siłka, J.; Wiczorek, M. Deep learning based crowd counting model for drone assisted systems. In Proceedings of the 4th ACM MobiCom Workshop on Drone Assisted Wireless Communications for 5G and Beyond, Virtual, 29 October 2021; pp. 31–36.
12. Boutros, F.; Siebke, P.; Klemm, M.; Damer, N.; Kirchbuchner, F.; Kuijper, A. PocketNet: Extreme lightweight face recognition network using neural architecture search and multistep knowledge distillation. *IEEE Access* **2022**, *10*, 46823–46833. [[CrossRef](#)]
13. Feng, Y.; Wang, H.; Hu, H.R.; Yu, L.; Wang, W.; Wang, S. Triplet distillation for deep face recognition. In Proceedings of the 2020 IEEE International Conference on Image Processing (ICIP), Abu Dhabi, United Arab Emirates, 25–28 October 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 808–812.
14. Luo, J.H.; Wu, J.; Lin, W. Thinet: A filter level pruning method for deep neural network compression. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 5058–5066.
15. Liu, J.; Zhuang, B.; Zhuang, Z.; Guo, Y.; Huang, J.; Zhu, J.; Tan, M. Discrimination-aware network pruning for deep model compression. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *44*, 4035–4051. [[CrossRef](#)] [[PubMed](#)]
16. Choukroun, Y.; Kravchik, E.; Yang, F.; Kisilev, P. Low-bit quantization of neural networks for efficient inference. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), Seoul, Republic of Korea, 27–28 October 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 3009–3018.
17. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4510–4520.
18. Ma, N.; Zhang, X.; Zheng, H.T.; Sun, J. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 116–131.
19. Han, K.; Wang, Y.; Tian, Q.; Guo, J.; Xu, C.; Xu, C. Ghostnet: More features from cheap operations. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 1580–1589.
20. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.
21. Howard, A.; Sandler, M.; Chu, G.; Chen, L.C.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; et al. Searching for mobilenetv3. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 1314–1324.
22. Zhang, X.; Zhou, X.; Lin, M.; Sun, J. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 6848–6856.
23. Wiczorek, M.; Siłka, J.; Woźniak, M.; Garg, S.; Hassan, M.M. Lightweight Convolutional Neural Network Model for Human Face Detection in Risk Situations. *IEEE Trans. Ind. Inform.* **2022**, *18*, 4820–4829. [[CrossRef](#)]
24. Chen, S.; Liu, Y.; Gao, X.; Han, Z. Mobilefacenets: Efficient cnns for accurate real-time face verification on mobile devices. In *Chinese Conference on Biometric Recognition*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 428–438.
25. Martınez-Dıaz, Y.; Luevano, L.S.; Mendez-Vazquez, H.; Nicolas-Dıaz, M.; Chang, L.; Gonzalez-Mendoza, M. Shufflefacenet: A lightweight face architecture for efficient and highly-accurate face recognition. In Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, Seoul, Republic of Korea, 27–28 October 2019.
26. Yan, M.; Zhao, M.; Xu, Z.; Zhang, Q.; Wang, G.; Su, Z. Vargfacenet: An efficient variable group convolutional neural network for lightweight face recognition. In Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, Seoul, Republic of Korea, 27–28 October 2019.
27. Duong, C.N.; Quach, K.G.; Jalata, I.; Le, N.; Luu, K. Mobiface: A lightweight deep learning face recognition on mobile devices. In Proceedings of the 2019 IEEE 10th International Conference on Biometrics Theory, Applications and Systems (BTAS), Tampa, FL, USA, 23–26 September 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1–6.

28. Taigman, Y.; Yang, M.; Ranzato, M.; Wolf, L. Deepface: Closing the gap to human-level performance in face verification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 1701–1708.
29. Sun, Y.; Wang, X.; Tang, X. Deep learning face representation from predicting 10,000 classes. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 1891–1898.
30. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
31. Sun, Y.; Chen, Y.; Wang, X.; Tang, X. Deep learning face representation by joint identification-verification. *Adv. Neural Inf. Process. Syst.* **2014**, 27.
32. Schroff, F.; Kalenichenko, D.; Philbin, J. Facenet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 815–823.
33. Liu, J.; Deng, Y.; Bai, T.; Wei, Z.; Huang, C. Targeting ultimate accuracy: Face recognition via deep embedding. *arXiv* **2015**, arXiv:1506.07310.
34. Sun, Y.; Wang, X.; Tang, X. Sparsifying neural network connections for face recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 4856–4864.
35. Wang, F.; Xiang, X.; Cheng, J.; Yuille, A.L. Normface: L2 hypersphere embedding for face verification. In Proceedings of the 25th ACM international conference on Multimedia, Mountain View, CA, USA, 23–27 October 2017; pp. 1041–1049.
36. Ranjan, R.; Castillo, C.D.; Chellappa, R. L2-constrained softmax loss for discriminative face verification. *arXiv* **2017**, arXiv:1703.09507.
37. Liu, Y.; Li, H.; Wang, X. Rethinking feature discrimination and polymerization for large-scale recognition. *arXiv* **2017**, arXiv:1710.00870.
38. Sun, Y.; Cheng, C.; Zhang, Y.; Zhang, C.; Zheng, L.; Wang, Z.; Wei, Y. Circle loss: A unified perspective of pair similarity optimization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 6398–6407.
39. Zhang, X.; Zhao, R.; Qiao, Y.; Wang, X.; Li, H. Adacos: Adaptively scaling cosine logits for effectively learning deep face representations. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 10823–10832.
40. Liu, H.; Zhu, X.; Lei, Z.; Li, S.Z. Adaptiveface: Adaptive margin and sampling for face recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–17 June 2019; pp. 11947–11956.
41. Huang, Y.; Wang, Y.; Tai, Y.; Liu, X.; Shen, P.; Li, S.; Li, J.; Huang, F. Curricularface: Adaptive curriculum learning loss for deep face recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 5901–5910.
42. Kim, I.; Han, S.; Park, S.J.; Baek, J.W.; Shin, J.; Han, J.J.; Choi, C. Discface: Minimum discrepancy learning for deep face recognition. In Proceedings of the Asian Conference on Computer Vision, Kyoto, Japan, 30 November–4 December 2020.
43. Wu, X.; He, R.; Sun, Z.; Tan, T. A light CNN for deep face representation with noisy labels. *IEEE Trans. Inf. Forensics Secur.* **2018**, 13, 2884–2896. [[CrossRef](#)]
44. He, K.; Zhang, X.; Ren, S.; Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–15 December 2015; pp. 1026–1034.
45. Nair, V.; Hinton, G.E. Rectified linear units improve restricted boltzmann machines. In Proceedings of the Icml, Haifa, Israel, 21–24 June 2010.
46. Cao, Q.; Shen, L.; Xie, W.; Parkhi, O.M.; Zisserman, A. Vggface2: A dataset for recognising faces across pose and age. In Proceedings of the 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018), Xi'an, China, 15–19 May 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 67–74.
47. Guo, Y.; Zhang, L.; Hu, Y.; He, X.; Gao, J. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In *European Conference on Computer Vision*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 87–102.
48. Huang, G.B.; Mattar, M.; Berg, T.; Learned-Miller, E. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. In *Workshop on Faces in 'Real-Life' Images: Detection, Alignment, And Recognition*; HAL Inria: Paris, France, 2008.
49. Zheng, T.; Deng, W. Cross-pose lfw: A database for studying cross-pose face recognition in unconstrained environments. *Beijing Univ. Posts Telecommun. Tech. Rep.* **2018**, 5, 7.
50. Zheng, T.; Deng, W.; Hu, J. Cross-age lfw: A database for studying cross-age face recognition in unconstrained environments. *arXiv* **2017**, arXiv:1708.08197.
51. Sengupta, S.; Chen, J.C.; Castillo, C.; Patel, V.M.; Chellappa, R.; Jacobs, D.W. Frontal to profile face verification in the wild. In Proceedings of the 2016 IEEE Winter Conference on Applications of Computer Vision (WACV), Lake Placid, NY, USA, 7–10 March 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 1–9.
52. Whitelam, C.; Taborsky, E.; Blanton, A.; Maze, B.; Adams, J.; Miller, T.; Kalka, N.; Jain, A.K.; Duncan, J.A.; Allen, K.; et al. Iarpa janus benchmark-b face dataset. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Honolulu, HI, USA, 21–26 July 2017; pp. 90–98.

53. Maze, B.; Adams, J.; Duncan, J.A.; Kalka, N.; Miller, T.; Otto, C.; Jain, A.K.; Niggel, W.T.; Anderson, J.; Cheney, J.; et al. Iarpa janus benchmark-c: Face dataset and protocol. In Proceedings of the 2018 International Conference on Biometrics (ICB), Gold Coast, Australia, 20–23 February 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 158–165.
54. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. Pytorch: An imperative style, high-performance deep learning library. *Adv. Neural Inf. Process. Syst.* **2019**, 32.
55. Li, X.; Wang, F.; Hu, Q.; Leng, C. Airface: Lightweight and efficient model for face recognition. In Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, Seoul, Republic of Korea, October 2019.
56. Zhang, K.; Zhang, Z.; Li, Z.; Qiao, Y. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Process. Lett.* **2016**, 23, 1499–1503. [[CrossRef](#)]

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.