NormFace: L_2 Hypersphere Embedding for Face Verification

Feng Wang*
University of Electronic Science and Technology of China
2006 Xiyuan Ave.
Chengdu, Sichuan 611731
feng.wff@gmail.com

Xiang Xiang
Johns Hopkins University
3400 N. Charles St.
Baltimore, Maryland 21218
xxiang@cs.jhu.edu

Jian Cheng
University of Electronic Science and Technology of China
2006 Xiyuan Ave.
Chengdu, Sichuan 611731
chengjian@uestc.edu.cn

Alan L. Yuille
Johns Hopkins University
3400 N. Charles St.
Baltimore, Maryland 21218
alan.yuille@jhu.edu

ABSTRACT

Thanks to the recent developments of Convolutional Neural Networks, the performance of face verification methods has increased rapidly. In a typical face verification method, feature normalization is a critical step for boosting performance. This motivates us to introduce and study the effect of normalization during training. But we find this is non-trivial, despite normalization being differentiable. We identify and study four issues related to normalization through mathematical analysis, which yields understanding and helps with parameter settings. Based on this analysis we propose two strategies for training using normalized features. The first is a modification of softmax loss, which optimizes cosine similarity instead of inner-product. The second is a reformulation of metric learning by introducing an agent vector for each class. We show that both strategies, and small variants, consistently improve performance by between 0.2% to 0.4% on the LFW dataset based on two models. This is significant because the performance of the two models on LFW dataset is close to saturation at over 98%.

CCS CONCEPTS

• **Computing methodologies → Object identification**; **Supervised learning by classification**; **Neural networks**; Regularization;

KEYWORDS

Face Verification, Metric Learning, Feature Normalization

1 INTRODUCTION

In recent years, Convolutional neural networks (CNNs) achieve state-of-the-art performance for various computer vision tasks, such as object recognition [12, 29, 32], detection [5], segmentation [19] and so on. In the field of face verification, CNNs have already surpassed humans' abilities on several benchmarks[20, 33].

The most common pipeline for a face verification application involves face detection, facial landmark detection, face alignment, feature extraction, and finally feature comparison. In the feature comparison step, the cosine similarity or equivalently L_2 normalized Euclidean distance is used to measure the similarities between features. The cosine similarity $\frac{\langle \cdot, \cdot \rangle}{\|\cdot\|\|\cdot\|}$ is a similarity measure which is independent of magnitude. It can be seen as the normalized version of inner-product of two vectors. But in practice the inner product without normalization is the most widely-used similarity measure when training a CNN classification models [12, 29, 32]. In other words, the similarity or distance metric used during training is different from that used in the testing phase. To our knowledge, no researcher in the face verification community has clearly explained why the features should be normalized to calculate the similarity in the testing phase. Feature normalization is treated only as a trick to promote the performance during testing.

To illustrate this, we performed an experiment which compared the face features without normalization, *i.e.* using the unnormalized inner-product or Euclidean distance as the similarity measurement. The features were extracted from an online available model [36][1]. We followed the standard protocol of *unrestricted with labeled outside data*[9] and test the model on the Labeled Faces in the Wild (LFW) dataset[10]. The results are listed in Table 1.

Table 1: Effect of Feature Normalization

Similarity	Before Normalization	After Normalization
Inner-Product	98.27%	98.98%
Euclidean	98.35%	98.95%

As shown in the table, feature normalization promoted the performance by about 0.6% ∼ 0.7%, which is a significant improvement since the accuracies are already above 98%. Feature normalization seems to be a crucial step to get good performance during testing. Noting that the normalization operation is differentiable, there is no reason that stops us importing this operation into the CNN model to perform end-to-end training.

*Alan L. Yuille's visiting student.

[1] https://github.com/ydwen/caffe-face

MM '17, October 23–27, 2017, Mountain View, CA, USA.
© 2017 ACM. ISBN 978-1-4503-4906-2/17/10…$15.00
DOI: https://doi.org/10.1145/3123266.3123359

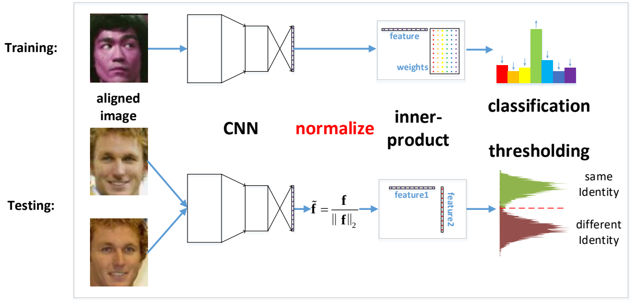


Figure 1: Pipeline of face verification model training and testing using a classification loss function. Previous works did not use the normalization after feature extraction during training. But in the testing phase, all methods used a normalized similarity, e.g. cosine, to compare two features.

Some previous works[23, 28] successfully trained CNN models with the features being normalized in an end-to-end fashion. However, both of them used the triplet loss, which needs to sample triplets of face images during training. It is difficult to train because we usually need to implement hard mining algorithms to find non-trivial triplets[28]. Another route is to train a classification network using softmax loss[31, 38] and regularizations to limit the intra-class variance[16, 36]. Furthermore, some works combine the classification and metric learning loss functions together to train CNN models[31, 41]. All these methods that used classification loss functions, e.g. softmax loss, did not apply feature normalization, even though they all used normalized similarity measure, e.g. cosine similarity, to get the confidence of judging two samples being of the same identity at testing phase(Figure 1).

We did an experiment by normalizing both the features and the weights of the last inner-product layer to build a cosine layer in an ordinary CNN model. After sufficient iterations, the network still did not converge. After observing this phenomenon, we deeply dig into this problem. In this paper, we will find out the reason and propose methods to enable us to train the normalized features.

To sum up, in this work, we analyze and answer the questions mentioned above about the feature normalization and the model training:

(1) Why is feature normalization so efficient when comparing the CNN features trained by classification loss, especially for softmax loss?
(2) Why does directly optimizing the cosine similarity using softmax loss cause the network to fail to converge?
(3) How to optimize a cosine similarity when using softmax loss?
(4) Since models with softmax loss fail to converge after normalization, are there any other loss functions suitable for normalized features?

For the first question, we explain it through a property of softmax loss in Section 3.1. For the second and third questions, we provide a bound to describe the difficulty of using softmax loss to optimize a cosine similarity and propose using the scaled cosine similarity in Section 3.3. For the fourth question, we reformulate a set of loss functions in metric learning, such as contrastive loss and triplet loss to perform the classification task by introducing an 'agent' strategy (Section 4). Utilizing the 'agent' strategy, there is no need to sample pairs and triplets of samples nor to implement the hard mining algorithm.

We also propose two tricks to improve performance for both static and video face verification. The first is to merge features extracted from both original image and mirror image by summation, while previous works usually merge the features by concatenation[31, 36]. The second is to use histogram of face similarities between video pairs instead of the mean[23, 36] or max[39] similarity when making classification.

Finally, by experiments, we show that normalization during training can promote the accuracies of two publicly available state-of-the-art models by 0.2 ∼ 0.4% on LFW[10] and about 0.6% on YTF[37].

2 RELATED WORKS

Normalization in Neural Network. Normalization is a common operation in modern neural network models. Local Response Normalization and Local Contrast Normalization are studied in the AlexNet model[12], even though these techniques are no longer common in modern models. Batch normalization[11] is widely used to accelerate the speed of neural network convergence by reducing the internal covariate shift of intermediate features. Weight normalization [27] was proposed to normalize the weights of convolution layers and inner-product layers, and also lead to faster convergence speed. Layer normalization [1] tried to solve the batch size dependent problem of batch normalization, and works well on Recurrent Neural Networks.

Face Verification. Face verification is to decide whether two images containing faces represent the same person or two different people, and thus is important for access control or re-identification tasks. Face verification using deep learning techniques achieved a series of breakthroughs in recent years [20, 23, 28, 33, 36]. There are mainly two types of methods according to their loss functions. One type uses metric learning loss functions, such as contrastive loss[4, 40] and triplet loss[23, 28, 34]. The other type uses softmax loss and treats the problem as a classification task, but also constrains the intra-class variance to get better generalization for comparing face features [16, 36]. Some works also combine both kinds of loss functions[40, 41].

Metric Learning. Metric learning[4, 25, 34] tries to learn semantic distance measures and embeddings such that similar samples are nearer and different samples are further apart from each other on a manifold. With the help of neural networks' enormous ability of representation learning, deep metric learning[3, 19] can do even better than the traditional methods. Recently, more complicated loss functions were proposed to get better local embedding structures[8, 22, 30].

Recent Works on Normalization. Recently, cosine similarity [17] was used instead of the inner-product for training a CNN for person recognition, which is quite similar with face verification. The Cosine Loss proposed in [17] is quite similar with the one described in Section 3.3, normalizing both the features and weights. L2-softmax[24] shares a similar analysis about the convergence problem described in Section 3.3. In [24], the authors also propose to add a scale parameter after normalization, but they only normalize the features. SphereFace[35] improves the performance of Large

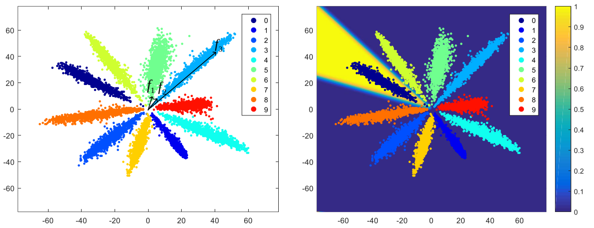


Figure 2: *Left:* The optimized 2-dimensional feature distribution using softmax loss on MNIST[14] dataset. Note that the Euclidean distance between f_1 and f_2 is much smaller than the distance between f_2 and f_3, even though f_2 and f_3 are from the same class. *Right:* The softmax probability for class 0 on the 2-dimension plane. Best viewed in color.

Margin Softmax[16] by normalizing the weights of the last inner-product layer only. Von Mises-Fisher Mixture Model(vMFMM)[21] interprets the hypersphere embedding as a mixture of von Mises-Fisher distributions. To sum up, the Cosine Loss[17], vMFMM[21] and our proposed loss functions optimize both features and weights, while the L2-softmax[24] normalizes the features only and the SphereFace[35] normalizes the weights only.

3 L_2 NORMALIZATION LAYER

In this section, we answer the question why we should normalize the features when the loss function is softmax loss and why the network does not converge if we directly put a softmax loss on the normalized features.

3.1 Necessity of Normalization

In order to give an intuitive feeling about the softmax loss, we did a toy experiment of training a deeper LeNet[13] model on the MNIST dataset[14]. We reduced the number of the feature dimension to 2 and plot 10,000 2-dimensional features from the training set on a plane in Figure 2. From the figure, we find that $\mathbf{f}_2$ can be much closer to $\mathbf{f}_1$ than to $\mathbf{f}_3$ if we use Euclidean distance as the metric. Hence directly using the features for comparison may lead to bad performance. At the same time, we find that the angles between feature vectors seem to be a good metric compared with Euclidean distance or inner-product operations. Actually, most previous work takes the cosine of the angle between feature vectors as the similarity [31, 36, 38], even though they all use softmax loss to train the network. Since the most common similarity metric for softmax loss is the inner-product with unnormalized features, there is a gap between the metrics used in the training and testing phases.

The reason why the softmax loss tends to create a 'radial' feature distribution (Figure 2) is that the softmax loss actually acts as the *soft* version of *max* operator. Scaling the feature vectors' magnitude does not affect the assignment of its class. Formally speaking, we recall the definition of the softmax loss,

$$\mathcal{L}_S = -\frac{1}{m}\sum_{i=1}^{m} log \frac{e^{W_{y_i}^T \mathbf{f}_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T \mathbf{f}_i + b_j}}, \tag{1}$$

where m is the number of training samples, n is the number of classes, $\mathbf{f}_i$ is the feature of the i-th sample, y_i is the corresponding label in range $[1, n]$, W and b are the weight matrix and the bias vector of the last inner-product layer before the softmax loss, W_j is the j-th column of W, which is corresponding to the j-th class. In the testing phase, we classify a sample by

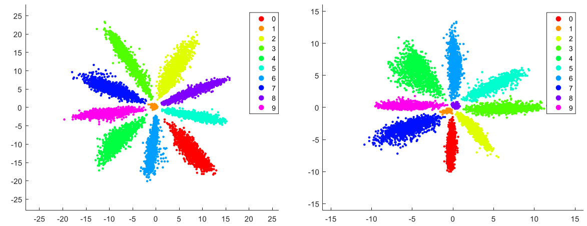


Figure 3: Two selected scatter diagrams when bias term is added after inner-product operation. Please note that there are one or two clusters that are located near the zero point. If we normalize the features of the center clusters, they would spread everywhere on the unit circle, which would cause misclassification. Best viewed in color.

$$Class(\mathbf{f}) = i = \arg\max_i (W_i^T \mathbf{f} + b_i). \tag{2}$$

In this case, we can infer that $(W_i\mathbf{f} + b_i) - (W_j\mathbf{f} + b_j) \geq 0, \forall j \in [1, n]$. Using this inequality, we obtain the following proposition.

Proposition 1. *For the softmax loss with **no-bias** inner-product similarity as its metric, let $P_i(\mathbf{f}) = \frac{e^{W_i^T \mathbf{f}}}{\sum_{j=1}^{n} e^{W_j^T \mathbf{f}}}$ denote the probability of $\mathbf{x}$ being classified as class i. For any given scale $s > 1$, if $i = \arg\max_j (W_j^T \mathbf{f})$, then $P_i(s\mathbf{f}) \geq P_i(\mathbf{f})$ always holds.*

The proof is given in Appendix 8.1. This proposition implies that softmax loss always encourages well-separated features to have bigger magnitudes. This is the reason why the feature distribution of softmax is 'radial'. However, we may not need this property as shown in Figure2. By normalization, we can eliminate its effect. Thus, we usually use the cosine of two feature vectors to measure the similarity of two samples.

However, Proposition 1 does not hold if a bias term is added after the inner-product operation. In fact, the weight vector of the two classes could be the same and the model still could make a decision via the biases. We found this kind of case during the MNIST experiments and the scatters are shown in Figure 3. It can be discovered from the figure that the points of some classes all locate around the zero point, and after normalization the points from each of these classes may be spread out on the unit circle, overlapping with other classes. In these cases, feature normalization may destroy the discrimination ability of the specific classes. To avoid this kind of risk, we do not add the bias term before the softmax loss in this work, even though it is commonly used for classification tasks.

3.2 Layer Definition

In this paper, we define $\|\mathbf{x}\|_2 = \sqrt{\sum_i \mathbf{x}_i^2 + \epsilon}$, where ϵ is a small positive value to prevent dividing zero. For an input vector $\mathbf{x} \in \mathcal{R}^n$,

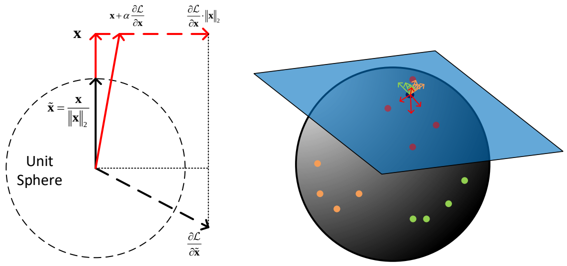

$\mathbf{x}+\alpha\frac{\partial\mathcal{L}}{\partial\mathbf{x}}$
$\frac{\partial\mathcal{L}}{\partial\mathbf{x}}\cdot\|\mathbf{x}\|_2$
$\mathbf{x}$
$\tilde{\mathbf{x}}=\frac{\mathbf{x}}{\|\mathbf{x}\|_2}$
Unit Sphere
$\frac{\partial\mathcal{L}}{\partial\tilde{\mathbf{x}}}$

Figure 4: *Left:* The normalization operation and its gradient in 2-dimensional space. Please note that $\|\mathbf{x}+\alpha\frac{\partial\mathcal{L}}{\partial\mathbf{x}}\|$ is always bigger than $\|\mathbf{x}\|$ for all $\alpha > 0$ because of the Pythagoras theorem. *Right:* An example of the gradients w.r.t. the weight vector. All the gradients are in the tangent space of the unit sphere (denoted as the blue plane). The red, yellow and green points are normalized features from 3 different classes. The blue point is the normalized weight corresponding to the red class. Here we assume that the model tries to make features get close to their corresponding classes and away from other classes. Even though we illustrate the gradients applied on the normalized weight only, please note that opposite gradients are also applied on the normalized features (red, yellow, green points). Finally, all the gradients are accumulated together to decide which direction the weight should be updated. Best viewed in color, zoomed in.

an L_2 normalization layer outputs the normalized vector,

$$\tilde{\mathbf{x}} = \frac{\mathbf{x}}{\|\mathbf{x}\|_2} = \frac{\mathbf{x}}{\sqrt{\sum_i \mathbf{x}_i^2 + \epsilon}}. \tag{3}$$

Here $\mathbf{x}$ can be either the feature vector $\mathbf{f}$ or one column of the weight matrix W_i. In backward propagation, the gradient w.r.t. $\mathbf{x}$ can be obtained by the chain-rule,

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\mathbf{x}_i} &= \frac{\partial\mathcal{L}}{\partial\tilde{\mathbf{x}}_i}\frac{\partial\tilde{\mathbf{x}}_i}{\partial\mathbf{x}_i} + \sum_j \frac{\partial\mathcal{L}}{\partial\tilde{\mathbf{x}}_j}\frac{\partial\tilde{\mathbf{x}}_j}{\partial\|\mathbf{x}\|_2}\frac{\partial\|\mathbf{x}\|_2}{\partial\mathbf{x}_i} \\ &= \frac{\frac{\partial\mathcal{L}}{\partial\tilde{\mathbf{x}}_i} - \tilde{\mathbf{x}}_i\sum_j\frac{\partial\mathcal{L}}{\partial\tilde{\mathbf{x}}_j}\tilde{\mathbf{x}}_j}{\|\mathbf{x}\|_2}.\end{aligned} \tag{4}$$

It is noteworthy that vector $\mathbf{x}$ and $\frac{\partial\mathcal{L}}{\partial\mathbf{x}}$ are orthogonal with each other, *i.e.* $\langle\mathbf{x}, \frac{\partial\mathcal{L}}{\partial\mathbf{x}}\rangle = 0$. From a geometric perspective, the gradient $\frac{\partial\mathcal{L}}{\partial\mathbf{x}}$ is the projection of $\frac{\partial\mathcal{L}}{\partial\tilde{\mathbf{x}}}$ onto the tangent space of the unit hypersphere at normal vector $\tilde{\mathbf{x}}$ (see Figure 4). From Figure 4 left, it can be inferred that after update, $\|\mathbf{x}\|_2$ always increases. In order to prevent $\|\mathbf{x}\|_2$ growing infinitely, weight decay is necessary on vector $\mathbf{x}$.

3.3 Reformulating Softmax Loss

Using the normalization layer, we can directly optimize the cosine similarity,

$$d(\mathbf{f}, \mathbf{W_i}) = \frac{\langle\mathbf{f}, \mathbf{W_i}\rangle}{\|\mathbf{f}\|_2\|\mathbf{W_i}\|_2}, \tag{5}$$

where $\mathbf{f}$ is the feature and $\mathbf{W_i}$ represents the i-th column of the weight matrix of the inner-product layer before softmax loss layer. However, after normalization, the network fails to converge. The loss only decreases a little and then converges to a very big value within a few thousands of iterations. After that the loss does not decrease no matter how many iterations we train and how small the learning rate is.

This is mainly because the range of $d(\mathbf{f}, \mathbf{W_i})$ is only $[-1, 1]$ after normalization, while it is usually between $(-20, 20)$ and $(-80, 80)$ when we use an inner-product layer and softmax loss. This low range problem may prevent the probability $P_{y_i}(\mathbf{f};\mathbf{W}) = \frac{e^{\mathbf{W}_{y_i}^T\mathbf{f}}}{\sum_j^n e^{\mathbf{W}_j^T\mathbf{f}}}$, where y_i is $\mathbf{f}$'s label, from getting close to 1 even when the samples are well-separated. In the extreme case, $\frac{e^1}{e^1+(n-1)e^{-1}}$ is very small (0.45 when $n = 10$; 0.007 when $n = 1000$), even though in this condition the samples of all other classes are on the other side of the unit hypersphere. Since the gradient of softmax loss w.r.t. the ground truth label is $1 - P_{y_i}$, the model will always try to give large gradients to the well separated samples, while the harder samples may not get sufficient gradients.

To better understand this problem, we give a bound to clarify how small the softmax loss can be in the best case.

Proposition 2. (Softmax Loss Bound After Normalization) *Assume that every class has the same number of samples, and all the samples are well-separated,* i.e. *each sample's feature is exactly same with its corresponding class's weight. If we normalize both the features and every column of the weights to have a norm of ℓ, the softmax loss will have a lower bound,* $\log\left(1 + (n-1)\,e^{-\frac{n}{n-1}\ell^2}\right)$*, where n is the class number.*

The proof is given in Appendix 8.2. Even though reading the proof need patience, we still encourage readers to read it because you may get better understanding about the hypersphere manifold from it.

This bound implies that if we just normalize the features and weights to 1, the softmax loss will be trapped at a very high value on training set, even if no regularization is applied. For a real example, if we train the model on the CASIA-Webface dataset ($n = 10575$), the loss will decrease from about 9.27 to about 8.50. The bound for this condition is 8.27, which is very close to the real value. This suggests that our bound is very tight. To give an intuition for the bound, we also plot the curve of the bound as a function of the norm ℓ in Figure 5.

After we obtain the bound, the solution to the convergence problem is clear. By normalizing the features and columns of weight to a bigger value ℓ instead of 1, the softmax loss can continue to decrease. In practice, we may implement this by directly appending a scale layer after the cosine layer. The scale layer has only one learnable parameter $s = \ell^2$. We may also fix it to a value that is large enough referring to Figure 5, say 20 or 30 for different class number. However, we prefer to make the parameter automatically learned by back-propagation instead of introducing a new hyper-parameter for elegance. Finally, the softmax loss with cosine distance is defined as

$$\mathcal{L}_{S'} = -\frac{1}{m}\sum_{i=1}^{m} log\frac{e^{s\tilde{W}_{y_i}^T\tilde{\mathbf{f}}_i}}{\sum_{j=1}^{n} e^{s\tilde{W}_j^T\tilde{\mathbf{f}}_i}}, \tag{6}$$

where $\tilde{\mathbf{x}}$ is the normalized $\mathbf{x}$.

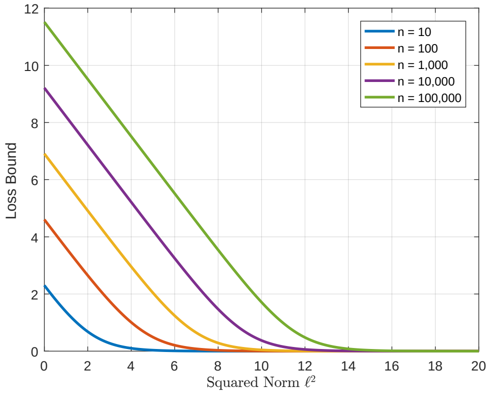


Figure 5: The softmax loss' lower bound as a function of features and weights' norm. Note that the x axis is the squared norm ℓ^2 because we add the scale parameter directly on the cosine distance in practice.

4 REFORMULATING METRIC LEARNING

Metric Learning, or specifically deep metric learning in this work, usually takes pairs or triplets of samples as input, and outputs the distance between them. In deep metric models, it is a common strategy to normalize the final features[22, 23, 28]. It seems that normalization does not cause any problems for metric learning loss functions. However, metric learning is more difficult to train than classification because the possible input pairs or triplets in metric learning models are very large, namely $O(N^2)$ combinations for pairs and $O(N^3)$ combinations for triplets, where N is the amount of training samples. It is almost impossible to deal with all possible combinations during training, so sampling and hard mining algorithms are usually necessary[28], which are tricky and time-consuming. By contrast, in a classification task, we usually feed the data iteratively into the model, namely the input data is in order of $O(N)$. In this section, we attempt to reformulate some metric learning loss functions to do the classification task, while keeping their compatibility with the normalized features.

The most widely used metric learning methods in the face verification community are the contrastive loss[31, 40],

$$\mathcal{L}_C = \begin{cases} \|\tilde{\mathbf{f}}_i - \tilde{\mathbf{f}}_j\|_2^2, & c_i = c_j \\ \max(0, m - \|\tilde{\mathbf{f}}_i - \tilde{\mathbf{f}}_j\|_2^2), & c_i \neq c_j \end{cases}, \tag{7}$$

and the triplet loss[23, 28],

$$\mathcal{L}_{\mathcal{T}} = max(0, m + \|\tilde{\mathbf{f}}_i - \tilde{\mathbf{f}}_j\|_2^2 - \|\tilde{\mathbf{f}}_i - \tilde{\mathbf{f}}_k\|_2^2), \quad c_i = c_j, c_i \neq c_k, \tag{8}$$

where the two m's are the margins. Both of the two loss functions optimize the normalized Euclidean distance between feature pairs. Note that after normalization, the reformulated softmax loss can also be seen as optimizing the normalized Euclidean distance,

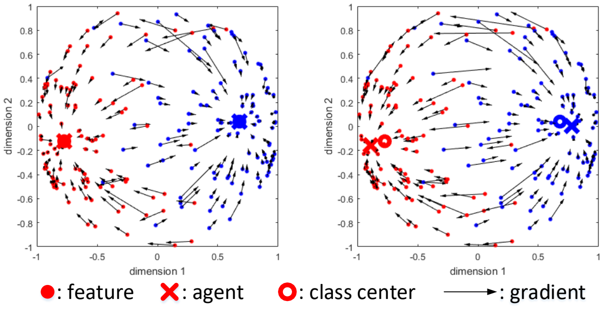


Figure 6: Illustration of how the C-contrastive loss works with two classes on a 3-d sphere (projected on a 2-d plane). *Left:* The special case of $m = 0$. In this case, the agents are only influenced by features from their own classes. The agents will finally converge to the centers of their corresponding classes. *Right:* Normal case of $m = 1$. In this case, the agents are influenced by all the features in the same classes and other classes' features in the margin. Hence the agents are shifted away from the boundary of the two classes. The features will follow their agents through the intra-class term $\|\tilde{\mathbf{f}}_i - \tilde{W}_j\|_2^2, c_i = j$ as the gradients shown in the figure. Best viewed in color.

$$\begin{aligned} \mathcal{L}_{\mathcal{S}'} &= -\frac{1}{m}\sum_{i=1}^{m} log \frac{e^{s\tilde{W}_{y_i}^T\tilde{\mathbf{f}}_i}}{\sum_{j=1}^{n} e^{s\tilde{W}_j^T\tilde{\mathbf{f}}_i}} \\ &= -\frac{1}{m}\sum_{i=1}^{m} log \frac{e^{-\frac{s}{2}\|\tilde{\mathbf{f}}_i-\tilde{W}_{y_i}\|_2^2}}{\sum_{j=1}^{n} e^{-\frac{s}{2}\|\tilde{\mathbf{f}}_i-\tilde{W}_j\|_2^2}}, \end{aligned} \tag{9}$$

because $\|\tilde{\mathbf{x}} - \tilde{\mathbf{y}}\|_2^2 = 2 - 2\tilde{\mathbf{x}}^T\tilde{\mathbf{y}}$. Inspired by this formulation, we modify one of the features to be one column of a weight matrix $W \in \mathbb{R}^{d\times n}$, where d is the dimension of the feature and n is the class number. We call column W_i as the 'agent' of the i-th class. The weight matrix W can be learned through back-propagation just as the inner-product layer. In this way, we can get a classification version of the contrastive loss,

$$\mathcal{L}_{C'} = \begin{cases} \|\tilde{\mathbf{f}}_i - \tilde{W}_j\|_2^2, & c_i = j \\ \max(0, m - \|\tilde{\mathbf{f}}_i - \tilde{W}_j\|_2^2), & c_i \neq j \end{cases}, \tag{10}$$

and the triplet loss,

$$\mathcal{L}_{\mathcal{T}'} = max(0, m + \|\tilde{\mathbf{f}}_i - \tilde{W}_j\|_2^2 - \|\tilde{\mathbf{f}}_i - \tilde{W}_k\|_2^2), \quad c_i = j, c_i \neq k. \tag{11}$$

To distinguish these two loss functions from their metric learning versions, we call them *C-contrastive loss* and *C-triplet loss* respectively, denoting that these loss functions are designed for classification.

Intuitively, W_j acts as a summarizer of the features in j-th class. If all classes are well-separated by the margin, the W_j's will roughly correspond to the means of features in each class (Figure 6 left). In more complicated tasks, features of different classes may be overlapped with each other. Then the W_j's will be shifted away from the boundaries. The marginal features (hard examples) are

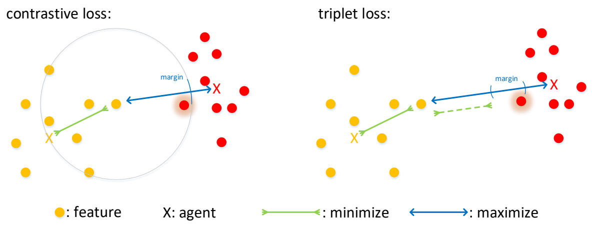


Figure 7: Classification version of contrastive loss (Left) and triplet loss (Right). The shadowed points are the marginal features that got omitted due to the 'agent' strategy. In the original version of the two losses, the shadowed points are also optimized. Best viewed in color.

guided to have bigger gradients in this case (Figure 6 right), which means they move further than easier samples during update.

However, there are some side effect of the agent strategy. After reformulation, some of the marginal features may not be optimized if we still use the same margin as the original version (Figure 7). Thus, we need larger margins to make more features get optimized. Mathematically, the error caused by the agent approximation is given by the following proposition.

Proposition 3. *Using an agent for each class instead of a specific sample would cause a distortion of* $\frac{1}{n_{C_i}}\sum_{j\in C_i}\left(d(f_0,f_j)-d(f_0,W_i)\right)^2$, *where* W_i *is the agent of the ith-class. The distortion is bounded by* $\frac{1}{n_{C_i}}\sum_{j\in C_i}d(f_j,W_i)^2$.

The proof is given in Appendix 8.3. This bound gives us a theoretical guidance of setting the margins. We can compute it on-the-fly during training using moving-average and display it to get better feelings about the progress. Empirically, the bound $\frac{1}{n_{C_i}}\sum_{j\in C_i}d(f_j,W_i)^2$ is usually $0.5\sim 0.6$. The recommendation value of the margins of the modified contrastive loss and triplet loss is 1 and 0.8 respectively.

Note that setting the margin used to be complicated work[40]. Following their work, we have to suspend training and search for a new margin for every several epochs. However, we no longer need to perform such a searching algorithm after applying normalization. Through normalization, the scale of features' magnitude is fixed, which makes it possible to fix the margin, too. In this strategy, we will not try to train models using the C-contrastive loss or the C-triplet loss without normalization because this is difficult.

5 EXPERIMENT

In this section, we first describe the experiment settings in Section 5.1. Then we evaluate our method on two different datasets with two different models in Section 5.2 and 5.3. Codes and models are released at https://github.com/happynear/NormFace.

5.1 Implementation Details

Baseline works. To verify our algorithm's universality, we choose two works as our baseline, Wu *et. al.*'s model [38][2] (Wu's model, for short) and Wen *et. al.*'s model [36][3] (Wen's model, for short). Wu's model is a 10-layer plain CNN with Maxout[6] activation unit. Wen's model is a 28-layer ResNet[7] trained with both softmax loss and center loss. Neither of these two models apply feature normalization or weight normalization. We strictly follow all the experimental settings as their papers, including the datasets[4], the image resolution, the pre-processing methods and the evaluation criteria.

Training. The proposed loss functions are appended after the feature layer, *i.e.* the second last inner-product layer. The features and columns of weight matrix are normalized to make their L_2 norm to be 1. Then the features and columns of the weight matrix are sent into a pairwise distance layer, *i.e.* inner-product layer to produce a cosine similarity or Euclidean distance layer to produce a normalized Euclidean distance. After calculating all the similarities or distances between each feature and each column, the proposed loss functions will give the final loss and gradients to the distances. The whole network models are trained end to end. To speed up the training procedure, we fine-tune the networks from the baseline models. Thus, a relatively small learning rate, say 1e-4 for Wu's model and 1e-3 for Wen's model, are applied to update the network through stochastic gradient descent (SGD) with momentum of 0.9.

Evaluation. Two datasets are utilized to evaluate the performance, one is Labeled Face in the Wild (LFW)[10] and another is Youtube Face (YTF)[37]. 10-fold validation is used to evaluate the performance for both datasets. After the training models converge, we continue to train them for $5,000$ iterations[5], during which we save a snapshot for every $1,000$ iterations. Then we run the evaluation codes on the five saved snapshots separately and calculate an average score to reduce disturbance. We extract features from both the frontal face and its mirror image and merge the two features by *element-wise summation*. Principle Component Analysis (PCA) is then applied on the training subset of the evaluation dataset to fit the features to the target domain. Similarity score is computed by the cosine distance of two sample's features after PCA. All the evaluations are based on the similarity scores of image pairs.

5.2 Experiments on LFW

The LFW dataset[10] contains $13,233$ images from $5,749$ identities, with large variations in pose, expression and illumination. All the images are collected from Internet. We evaluate our methods through two different protocols on LFW, one is the standard *unrestricted with labeled outside data* [9], which is evaluated on $6,000$ image pairs, and another is *BLUFR* [15] which utilize all $13,233$ images. It is noteworthy that there are three same identities in CASIA-Webface[40] and LFW[10]. We delete them during training to build a complete open-set validation.

We carefully test almost all combinations of the loss functions on the standard *unrestricted with labeled outside data* protocol. The results are listed in Table 2. Cosine similarity is used by *softmax + any* loss functions. The distance used by *C-contrastive* and *C-triplet* loss is the squared normalized Euclidean distance. The *C-triplet*

[2] https://github.com/AlfredXiangWu/face_verification_experiment

[3] https://github.com/ydwen/caffe-face

[4] Since the identity label of the Celebrity+[18] dataset is not publicly available, we follow Wen's released model which is trained on CASIA-Webface [40] only. Wu's model is also trained on CASIA-Webface [40] only.

[5] In each iteration we train 256 samples, i.e. the batch size is 256.

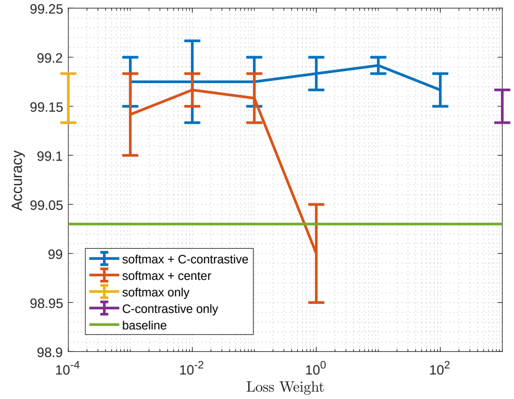


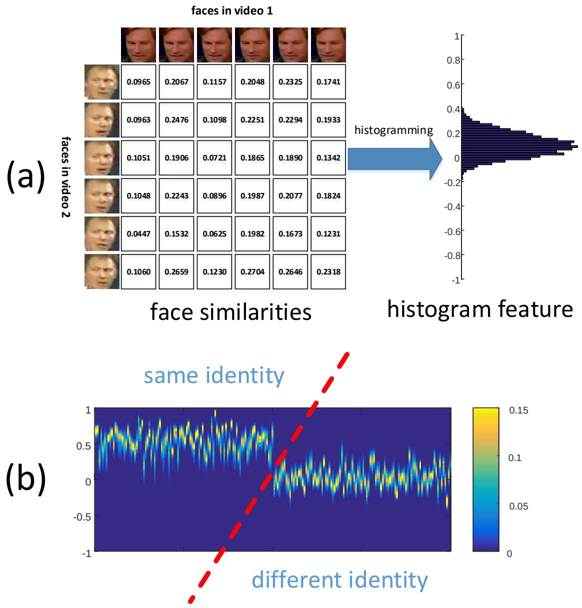
Figure 8: LFW accuracies as a function of the loss weight of C-contrastive loss or center loss with error bars. All these methods use the normalization strategy except for the baseline.

Table 2: Results on LFW 6,000 pairs using Wen's model[36]

loss function	Normalization	Accuracy
softmax	No	98.28%
softmax + dropout	No	98.35%
softmax + center[36]	No	99.03%
softmax	feature only	98.72%
softmax	weight only	98.95%
softmax	Yes	99.16% ± 0.025%
softmax + center	Yes	99.17% ± 0.017%
C-contrasitve	Yes	99.15% ± 0.017%
C-triplet	Yes	99.11% ± 0.008%
C-triplet + center	Yes	99.13% ± 0.017%
softmax + C-contrastive	Yes	99.19% ± 0.008%

+ *center* loss is implemented by forcing to optimize $\|\mathbf{x}_i - W_j\|_2^2$ even if $m + \|\mathbf{x}_i - W_j\|_2^2 - \|\mathbf{x}_i - W_k\|_2^2$ is less than 0. From Table 2 we can conclude that the loss functions have minor influence on the accuracy, and the normalization is the key factor to promote the performance. When combining the softmax loss with the C-contrastive loss or center loss, we need to add a hyper-parameter to make balance between the two losses. The highest accuracy, **99.2167**%, is obtained by softmax + 0.01 $*$ C-contrastive. However, pure softmax with normalization already works reasonably well.

We have also designed two ablation experiments of normalizing the features only or normalizing the columns of weight matrix only. During experiments we find that the scale parameter is necessary when normalizing the feature, while normalizing the weight does not need it. We cannot explain it so far. This is tricky but the network will collapse if the scale parameter is not properly added. From Table 2 we can conclude that normalizing the feature causes performance degradation, while normalizing the weight has little influence on the accuracy. Note that these two special cases of softmax loss are also fine-tuned based on Wen's model. When training from scratch, normalizing the weights only will cause the network collapse, while normalizing the features only will lead to a worse accuracy, 98.45%, which is better than the conventional softmax loss, but much worse than state-of-the-art loss functions.



0.0965	0.2067	0.1157	0.2048	0.2325	0.1741
0.0963	0.2476	0.1098	0.2251	0.2294	0.1933
0.1051	0.1906	0.0721	0.1865	0.1890	0.1342
0.1048	0.2243	0.0896	0.1987	0.2077	0.1824
0.0447	0.1532	0.0625	0.1982	0.1673	0.1231
0.1060	0.2659	0.1230	0.2704	0.2646	0.2318

Figure 9: (a): Illustration of how to generate a histogram feature for a pair of videos. We firstly create a pairwise score matrix by computing the cosine similarity between two face images from different video sequences. Then we accumulate all the scores in the matrix to create a histogram. (b): Visualization of histogram features extracted from 200 video pairs with both same identities and different identities. After collecting all histogram features, support vector machine (SVM) using histogram intersection kernel(HIK) is utilized to make a binary classification.

In Figure 8, we show the effect of the loss weights when using two loss functions. As shown in the figure, the C-contrastive loss is more robust to the loss weight. This is not surprising because C-contrastive loss can train a model by itself only, while the center loss, which only optimizes the intra-class variance, should be trained with other supervised losses together.

To make our experiment more convincing, we also train some of the loss functions on Wu's model[38]. The results are listed in Table 4. Note that in [38], Wu et. al. did not perform face mirroring when they evaluated their methods. In Table 4, we also present the result of their model after face mirroring and feature merging. As is shown in the table, the normalization operation still gives a significant boost to the performance.

On BLUFR protocol, the normalization technique works even better. Here we only compare some of the models with the baseline (Table 3). From Table 3 we can see that normalization could boost the performance significantly, which reveals that normalization technique could perform much better when the false alarm rate (FAR) is low.

Table 3: Results on LFW BLUFR[15] protocol

model	loss function	Normalization	TPR@FAR=0.1%	DIR@FAR=1%
ResNet	softmax + center[36]	No	93.35%	67.86%
ResNet	softmax	Yes	95.77%	73.92%
ResNet	C-triplet + center	Yes	95.73%	76.12%
ResNet	softmax + C-contrastive	Yes	95.83%	77.18%
MaxOut	softmax[38]	No	89.12%	61.79%
MaxOut	softmax	Yes	90.64%	65.22%
MaxOut	C-contrastive	Yes	90.32%	68.14%

Table 4: Results on LFW 6,000 pairs using Wu's model[38]

loss function	Normalization	Accuracy
softmax	No	98.13%
softmax + mirror	No	98.41%
softmax	Yes	98.75% ± 0.008%
C-contrastive	Yes	98.78% ± 0.017%
softmax + C-contrastive	Yes	98.71% ± 0.017%

Table 5: Results on YTF with Wen's model[36]

loss function	Normalization	Accuracy
softmax + center[36]	No	93.74%
softmax	Yes	94.24%
softmax + HIK-SVM	Yes	94.56%
C-triplet + center	Yes	94.3%
C-triplet + center + HIK-SVM	Yes	94.58%
softmax + C-contrastive	Yes	94.34%
softmax + C-contrastive + HIK-SVM	Yes	94.72%

5.3 Experiments on YTF

The YTF dataset[37] consists of 3,425 videos of 1,595 different people, with an average of 2.15 videos per person. We follow the *unrestricted with labeled outside data* protocol, which takes 5, 000 video pairs to evaluate the performance.

Previous works usually extract face features from all frames or some selected frames in a video. Then two videos can construct a confidence matrix C in which each element C_{ij} is the cosine distance of face features extracted from the i-th frame of the first video and j-th frame of the second video. The final score is computed by the average of all all elements in C. The one dimension score is then used to train a classifier, say SVM, to get the threshold of same identity or different identity.

Here we propose to use the histogram of elements in C as the feature to train the classifier. The bin of the histogram is set to 100 (Figure 9(a)). Then SVM with histogram intersection kernel (HIK-SVM)[2] is utilized to make a two-class classification (Figure 9(b)). This method encodes more information compared to the one dimensional mean value, and leads to better performance on video face verification.

The results are listed in Table 5. The models that perform better on LFW also show superior performance on YTF. Moreover, the newly proposed score histogram technique (HIK-SVM in the table) can improve the accuracy further by a significant gap.

6 CONCLUSION AND FUTURE WORK

In this paper, we propose to apply L_2 normalization operation on the features and the weight of the last inner-product layer when training a classification model. We explain the necessity of the normalization operation from both analytic and geometric perspective. Two kinds of loss functions are proposed to effectively train the normalized feature. One is a reformulated softmax loss with a scale layer inserted between the cosine score and the loss. Another is designed inspired by metric learning. We introduce an agent strategy to avoid the need of hard sample mining, which is a tricky and time-consuming work. Experiments on two different models both show superior performance over models without normalization. From three theoretical propositions, we also provide some guidance on the hyper-parameter setting, such as the bias term (Proposition 1), the scale parameter (Proposition 2) and the margin (Proposition 3).

Currently we can only *fine-tune* the network with normalization techniques based on other models. If we train a model with C-contrastive loss function, the final result is just as good as center loss[36]. But if we fine-tune a model, either Wen's model[36] or Wu's model[38], the performance could be further improved as shown in Table 2 and Table 4. More efforts are needed to find a way to train a model from scratch, while preserving at least a similar performance as fine-tuning.

Our methods and analysis in this paper are general. They can be used in other metric learning tasks, such as person re-identification or image retrieval. We will apply the proposed methods on these tasks in the future.

7 ACKNOWLEDGEMENT

This paper is funded by Office of Naval Research (N00014-15-1-2356), National Science Foundation (CCF-1317376), the National Natural Science Foundation of China (61671125, 61201271, 61301269) and the State Key Laboratory of Synthetical Automation for Process Industries (NO. PAL-N201401).

We thank Chenxu Luo and Hao Zhu for their assistance in formula derivation.

REFERENCES

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[2] Annalisa Barla, Francesca Odone, and Alessandro Verri. 2003. Histogram intersection kernel for image classification. In *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, Vol. 3. IEEE, III–513.

[3] Xinyuan Cai, Chunheng Wang, Baihua Xiao, Xue Chen, and Ji Zhou. 2012. Deep nonlinear metric learning with independent subspace analysis for face verification. In *ACM international conference on Multimedia*. ACM, 749–752.

[4] Sumit Chopra, Raia Hadsell, and Yann LeCun. 2005. Learning a similarity metric discriminatively, with application to face verification. In *IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 1. IEEE, 539–546.

[5] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jagannath Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*. 580–587.

[6] Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron C Courville, and Yoshua Bengio. 2013. Maxout Networks. *International Conference on Machine Learning* 28 (2013), 1319–1327.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.

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

[9] Gary B Huang and Erik Learned-Miller. 2014. Labeled faces in the wild: Updates and new reporting procedures. *Dept. Comput. Sci., Univ. Massachusetts Amherst, Amherst, MA, USA, Tech. Rep* (2014), 14–003.

[10] Gary B Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. 2007. *Labeled faces in the wild: A database for studying face recognition in unconstrained environments*. Technical Report. Technical Report 07-49, University of Massachusetts, Amherst.

[11] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).

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

[13] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[14] Yann LeCun, Corinna Cortes, and Christopher Burges. 1998. The mnist database of handwritten digits. (1998). http://yann.lecun.com/exdb/mnist/

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

[16] Weiyang Liu, Yandong Wen, Zhiding Yu, and Meng Yang. 2016. Large-Margin Softmax Loss for Convolutional Neural Networks. In *International Conference on Machine Learning*. 507–516.

[17] Yu Liu, Hongyang Li, and Xiaogang Wang. 2017. Learning Deep Features via Congenerous Cosine Loss for Person Recognition. *arXiv preprint arXiv:1702.06890* (2017).

[18] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Deep learning face attributes in the wild. In *Proceedings of the IEEE International Conference on Computer Vision*. 3730–3738.

[19] Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*. 3431–3440.

[20] Chaochao Lu and Xiaoou Tang. 2014. Surpassing human-level face verification performance on LFW with GaussianFace. *arXiv preprint arXiv:1404.3840* (2014).

[21] Jonathan Milgram StÃĺphane Gentric Liming Chen Md. Abul Hasnat, Julien BohnÃľ. 2017. von Mises-Fisher Mixture Model-based Deep learning: Application to Face Verification. *arXiv preprint arXiv:1706.04264* (2017).

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

[23] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. 2015. Deep Face Recognition.. In *BMVC*, Vol. 1. 6.

[24] Rajeev Ranjan, Carlos D. Castillo, and Rama Chellappa. 2017. L2-constrained Softmax Loss for Discriminative Face Verification. *arXiv preprint arXiv:1703.09507* (2017).

[25] Sam Roweis, Geoffrey Hinton, and Ruslan Salakhutdinov. 2004. Neighbourhood component analysis. *Advances in Neural Information Processing Systems* 17 (2004), 513–520.

[26] Walter Rudin and others. 1964. *Principles of mathematical analysis, Chapter 10*. Vol. 3. McGraw-Hill New York.

[27] Tim Salimans and Diederik P Kingma. 2016. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*. 901–901.

[28] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition*. 815–823.

[29] Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556* (2014).

[30] Kihyuk Sohn. 2016. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems*. 1849–1857.

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

[32] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*. 1–9.

[33] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. 2014. Deepface: Closing the gap to human-level performance in face verification. In *IEEE Conference on Computer Vision and Pattern Recognition*. 1701–1708.

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

[35] Zhiding Yu Ming Li Bhiksha Raj Weiyang Liu, Yandong Wen and Le Song. 2017. SphereFace: Deep Hypersphere Embedding for Face Recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

[36] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. 2016. A Discriminative Feature Learning Approach for Deep Face Recognition. In *European Conference on Computer Vision*. Springer, 499–515.

[37] Lior Wolf, Tal Hassner, and Itay Maoz. 2011. Face recognition in unconstrained videos with matched background similarity. In *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 529–534.

[38] Xiang Wu, Ran He, and Zhenan Sun. 2015. A Lightened CNN for Deep Face Representation. *arXiv preprint arXiv:1511.02683* (2015).

[39] Xiang Xiang and Trac D Tran. 2016. Pose-Selective Max Pooling for Measuring Similarity. *Lecture Notes in Computer Science* 10165 (2016).

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

[41] Xiao Zhang, Zhiyuan Fang, Yandong Wen, Zhifeng Li, and Yu Qiao. 2016. Range Loss for Deep Face Recognition with Long-tail. *arXiv preprint arXiv:1611.08976* (2016).

8 APPENDIX

8.1 Proof of Proposition 1

Proposition 1. *For the softmax loss with* ***no-bias*** *inner-product similarity as its metric, let* $P_i(\mathbf{f}) = \frac{e^{W_i^T\mathbf{f}}}{\sum_{j=1}^{n} e^{W_j^T\mathbf{f}}}$ *denote the probability of* $\mathbf{f}$ *being classified as class* i*. For a given scale* $s > 1$*, if* $i = \arg\max_j (W_j^T\mathbf{f})$*, then* $P_i(s\mathbf{f}) \geq P_i(\mathbf{f})$ *always holds.*

Proof: Let $t = s - 1$, after scaling, we have,

$$\begin{aligned} P_i(s\mathbf{f}) &= \frac{e^{W_i^T[(1+t)\mathbf{f}]}}{\sum_{j=1}^{n} e^{W_j^T[(1+t)\mathbf{f}]}} \\ &= \frac{e^{W_i^T\mathbf{f}}}{\sum_{j=1}^{n} e^{W_j^T\mathbf{f}+t(W_j^T\mathbf{f}-W_i^T\mathbf{f})}}. \end{aligned} \tag{12}$$

Recall that $W_i\mathbf{f} - W_j\mathbf{f} \geq 0$ if $i = \arg\max_j (W_j\mathbf{f})$, so $t(W_j^T\mathbf{f} - W_i^T\mathbf{f}) \leq 0$ always holds. Then

$$\begin{aligned} P_i(s\mathbf{f}) &\geq \frac{e^{W_i^T\mathbf{f}}}{\sum_{j=1}^{n} e^{W_j^T\mathbf{f}}} \\ &= P_i(\mathbf{f}). \end{aligned} \tag{13}$$

The equality holds if $W^T\mathbf{f} = \mathbf{0}$ or $W_i = W_j, \forall i,j \in [1,n]$, which is almost impossible in practice.

8.2 Proof of Proposition 2

Proposition 2. (Loss Bound After Normalization) *Assume that every class has the same number of samples, and all the samples are well-separated,* i.e. *each sample's feature is exactly the same with its corresponding class's weight. If we normalize both the features and every column of the weights to have a norm of* ℓ*, the softmax loss will have a lower bound,* $\log\left(1 + (n-1)\, e^{-\frac{n}{n-1}\ell^2}\right)$*, where* n *is the class number.*

Proof: Assume $\|W_i\| = \ell, \forall i \in [1,n]$ for convenience. Since we have already assumed that all samples are well-separated, we directly use W_i to represent the i-th class' feature.

The definition of the softmax loss is,

$$\mathcal{L}_{\mathcal{S}} = -\frac{1}{n}\sum_{i=1}^{n} \log \frac{e^{W_i^T W_i}}{\sum_{j=1}^{n} e^{W_i^T W_j}}. \tag{14}$$

This formula is different from Equation (1) because we assume that every class has the same sample number. By dividing $e^{W_i^T W_i} = e^{\ell^2}$ from both the numerator and denominator,

$$\begin{aligned} \mathcal{L}_{\mathcal{S}} &= -\frac{1}{n}\sum_{i=1}^{n} \log \frac{1}{1 + \sum_{j=1, j\neq i}^{n} e^{W_i^T W_j - \ell^2}} \\ &= \frac{1}{n}\sum_{i=1}^{n} \log\left(1 + \sum_{j=1, j\neq i}^{n} e^{W_i^T W_j - \ell^2}\right). \end{aligned} \tag{15}$$

Since $f(x) = e^x$ is a convex function, $\frac{1}{n}\sum_{i=1}^{n} e^{x_i} \geq e^{\frac{1}{n}\sum_{i=1}^{n} x_i}$, then we have,

$$\mathcal{L}_{\mathcal{S}} \geq \frac{1}{n}\sum_{i=1}^{n} \log\left(1 + (n-1) e^{\frac{1}{n-1}\sum_{j=1, j\neq i}^{n} (W_i^T W_j - \ell^2)}\right). \tag{16}$$

The equality holds if and only if all $W_i^T W_j, 1 \leq i < j \leq n$ have the same value, i.e., features from different classes have the same distance. Unfortunately, in d-dimension space, there are only $d+1$ unique vertices to ensure that every two vertices have the same distance. All these vertices will form a regular d-simplex[26], e.g., a regular 2-simplex is an equilateral triangle and a regular 3-simplex is a regular tetrahedron. Since the class number is usually much bigger than the dimension of feature in face verification datasets, this equality actually cannot hold in practice. One improvement over this inequality is taking the feature dimension into consideration because we actually have omitted the feature dimension term in this step.

Similar with $f(x) = e^x$, the softplus function $s(x) = \log(1 + Ce^x)$ is also a convex function when $C > 0$, so that $\frac{1}{n}\sum_{i=1}^{n} \log(1 + Ce^{x_i}) \geq \log(1 + Ce^{\frac{1}{n}\sum_{i=1}^{n} x_i})$, then we have

$$\begin{aligned} \mathcal{L}_{\mathcal{S}} &\geq \log\left(1 + (n-1)\, e^{\frac{1}{n(n-1)}\sum_{i=1}^{n}\sum_{j=1, j\neq i}^{n} (W_i^T W_j - \ell^2)}\right) \\ &= \log\left(1 + (n-1)\, e^{\left(\frac{1}{n(n-1)}\sum_{i=1}^{n}\sum_{j=1, j\neq i}^{n} W_i^T W_j\right) - \ell^2}\right). \end{aligned} \tag{17}$$

This equality holds if and only if $\forall W_i$, the sums of distances to other class' weight $\sum_{j=1, j\neq i}^{n} W_i^T W_j$ are all the same.

Note that

$$\|\sum_{i=1}^{n} W_i\|_2^2 = n\ell^2 + \sum_{i=1}^{n}\sum_{j=1, j\neq i}^{n} W_i^T W_j, \tag{18}$$

so

$$\sum_{i=1}^{n}\sum_{j=1, j\neq i}^{n} W_i^T W_j \geq -n\ell^2. \tag{19}$$

The equality holds if and only if $\sum_{i=1}^{n} W_i = \mathbf{0}$. Thus,

$$\begin{aligned} \mathcal{L}_{\mathcal{S}} &\geq \log\left(1 + (n-1)\, e^{-\frac{n\ell^2}{n(n-1)} - \ell^2}\right) \\ &= \log\left(1 + (n-1)\, e^{-\frac{n}{n-1}\ell^2}\right). \end{aligned} \tag{20}$$

8.3 Proof of Proposition 3

Proposition 3. *Using an agent for each class instead of a specific sample would cause a distortion of* $\frac{1}{n_{C_i}}\sum_{j\in C_i}\left(d(f_0, f_j) - d(f_0, W_i)\right)^2$*, where* W_i *is the agent of the* i*th-class. The distortion is bounded by* $\frac{1}{n_{C_i}}\sum_{j\in C_i} d(f_j, W_i)^2$*.*

Proof: Since d(x,y) is a metric, through the triangle inequality we have

$$d(f_0, W_i) - d(f_j, W_i) \leq d(f_0, f_j) \leq d(f_0, W_i) + d(f_j, W_i), \tag{21}$$

so

$$-d(f_j, W_i) \leq d(f_0, f_j) - d(f_0, W_i) \leq d(f_j, W_i), \tag{22}$$

and thus,

$$\left(d(f_0, f_j) - d(f_0, W_i)\right)^2 \leq d(f_j, W_i)^2. \tag{23}$$

As a result,

$$\frac{1}{n_{C_i}}\sum_{j\in C_i}\left(d(f_0, f_j) - d(f_0, W_i)\right)^2 \leq \frac{1}{n_{C_i}}\sum_{j\in C_i} d(f_j, W_i)^2. \tag{24}$$

8.4 Inference of Equation 4

Equation 4:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}_i} = \frac{\frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{x}}_i} - \tilde{\mathbf{x}}_i \sum_j \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{x}}_j} \tilde{\mathbf{x}}_j}{\|\mathbf{x}\|_2}. \tag{25}$$

Inference: Here we treat $\|\mathbf{x}\|_2$ as an independent variable. Note that $\tilde{\mathbf{x}} = \frac{\mathbf{x}}{\|\mathbf{x}\|_2}$ and $\|\mathbf{x}\|_2 = \sqrt{\sum_j \mathbf{x}_j^2 + \epsilon}$. We have,

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \mathbf{x}_i} &= \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{x}}_i} \frac{\partial \tilde{\mathbf{x}}_i}{\partial \mathbf{x}_i} + \sum_j \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{x}}_j} \frac{\partial \tilde{\mathbf{x}}_j}{\partial \|\mathbf{x}\|_2} \frac{\partial \|\mathbf{x}\|_2}{\partial \mathbf{x}_i} \\
&= \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{x}}} \frac{1}{\|\mathbf{x}\|_2} + \sum_j \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{x}}_j} \frac{-\tilde{\mathbf{x}}_j}{\|\mathbf{x}\|_2^2} \cdot \frac{1}{2} \frac{1}{\|\mathbf{x}\|_2} \cdot 2\mathbf{x}_i \\
&= \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{x}}} \frac{1}{\|\mathbf{x}\|_2} - \frac{\mathbf{x}_i}{\|\mathbf{x}\|_2} \sum_j \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{x}}_j} \frac{\tilde{\mathbf{x}}_j}{\|\mathbf{x}\|_2^2} \\
&= \frac{\frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{x}}_i} - \tilde{\mathbf{x}}_i \sum_j \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{x}}_j} \tilde{\mathbf{x}}_j}{\|\mathbf{x}\|_2}.
\end{aligned} \tag{26}$$

8.5 Proof of $\langle \mathbf{x}, \frac{\partial \mathcal{L}}{\partial \mathbf{x}} \rangle = 0$

Proof: The vectorized version of Equation 4 is

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}} = \frac{\frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{x}}} - \tilde{\mathbf{x}} \langle \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{x}}}, \tilde{\mathbf{x}} \rangle}{\|\mathbf{x}\|_2}. \tag{27}$$

So,

$$\begin{aligned}
\langle \mathbf{x}, \frac{\partial \mathcal{L}}{\partial \mathbf{x}} \rangle &= \frac{\langle \mathbf{x}, \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{x}}} \rangle - \langle \mathbf{x}, \tilde{\mathbf{x}} \rangle \langle \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{x}}}, \tilde{\mathbf{x}} \rangle}{\|\mathbf{x}\|_2} \\
&= \frac{\langle \mathbf{x}, \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{x}}} \rangle - \frac{\langle \mathbf{x}, \mathbf{x} \rangle \langle \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{x}}}, \mathbf{x} \rangle}{\|\mathbf{x}\|_2^2}}{\|\mathbf{x}\|_2} \\
&= \frac{\langle \mathbf{x}, \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{x}}} \rangle - \langle \tilde{\mathbf{x}}, \tilde{\mathbf{x}} \rangle \langle \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{x}}}, \mathbf{x} \rangle}{\|\mathbf{x}\|_2} \\
&= \frac{\langle \mathbf{x}, \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{x}}} \rangle - \langle \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{x}}}, \mathbf{x} \rangle}{\|\mathbf{x}\|_2} \\
&= 0.
\end{aligned} \tag{28}$$