

## HEATED-UP SOFTMAX EMBEDDING

Xu Zhang<sup>†</sup>, Felix Xinnan Yu<sup>‡</sup>, Svebor Karaman<sup>†</sup>, Wei Zhang<sup>†</sup>, & Shih-Fu Chang<sup>†</sup><sup>†</sup> Columbia University, <sup>‡</sup> Google Research

{xu.zhang, svebor.karaman, wz2363, sc250}@columbia.edu

felixyu@google.com

## ABSTRACT

**Metric learning** aims at learning a distance which is consistent with the semantic meaning of the samples. The problem is generally solved by learning an embedding for each sample such that the embeddings of samples of the same category are compact while the embeddings of samples of different categories are spread-out in the feature space. We study the features extracted from the second last layer of a deep neural network based classifier trained with the cross entropy loss on top of the softmax layer. We show that training classifiers with different temperature values of softmax function leads to features with different levels of compactness. Leveraging these insights, we propose a “heating-up” strategy to train a classifier with increasing temperatures, leading the corresponding embeddings to achieve state-of-the-art performance on a variety of metric learning benchmarks.

## 1 INTRODUCTION

Metric learning aims at learning a metric space in which the samples from the same classes are close (compact) and the samples from different classes are far away (spread-out in the space) (Hoffer & Ailon, 2015; Schroff et al., 2015; Zhang et al., 2017). It is a fundamental research topic in machine learning and has been widely explored in a variety of computer vision applications such as clustering (Xing et al., 2003; Ye et al., 2016), image retrieval (Lee et al., 2008; Zhang et al., 2017), face recognition (Guillaumin et al., 2009; Schroff et al., 2015), and person re-identification (Koestinger et al., 2012; Lisanti et al., 2017).

One solution for this problem is to define a loss function that enforces the properties of compactness and spread in the metric space. Two of the most popular loss functions are the contrastive loss (Chopra et al., 2005) and the triplet loss (Hoffer & Ailon, 2015). However, both losses face challenges in sampling, as usually there are a very large number of possible pairs or triplets in one dataset.

To overcome the sampling issue, a variety of hard mining strategies (Schroff et al., 2015; Mishchuk et al., 2017; Harwood et al., 2017) have been proposed. However, sampling the “hardest” samples (samples that most violate the predefine rules) often leads to poor local minima. Incorporating too many “easy” samples renders the training inefficient. Thus, designing a structured loss to perform hard-mining effectively and efficiently has become a hot research topic (Song et al., 2017; 2016).

Using features from the second last layer (a.k.a. bottleneck layer) of a deep neural network trained as a classifier with the softmax function and the cross-entropy loss works well for many metric learning based applications (Razavian et al., 2014) such as image retrieval (Babenko & Lempitsky, 2015) and face verification (Liu et al., 2017). However, the goals of classifier training and metric learning are different. The former aims at finding the best decision function while the latter is to learn an embedding such that embeddings of samples of the same category are compact while those of samples of different categories are “spread-out”. This motivates us to investigate the relation between metric learning and classifier training.

In this paper, we show that the temperature parameter in the softmax function, defined by Hinton et al. (2015) for knowledge transfer, plays an important role in determining the distribution of the embeddings from the bottleneck layer. Based on the observed relation, we propose to learn a classifier with an intermediate temperature at the beginning and increase the temperature during training. Compared to the state-of-the-art methods in deep metric learning, the proposed “heating-up” method

achieves significantly better performance for most cases and at least comparable performances for the rest.

The main contributions of this paper are that:

- we study the gradient of the softmax layer and show how the temperature parameter affects the final distribution of the embedding from the bottleneck layer;
- we propose a “heating-up” method that can be used to obtain an effective embedding with much better or at least comparable performance compared to state-of-the-art methods in deep metric learning.

## 2 RELATED WORKS

Siamese networks with contrastive loss (Chopra et al., 2005) was one of the earliest attempts to solve the metric learning problem. By sampling either two data samples from the same category (positive pair) or two different categories (negative pair), contrastive loss tries to pull two points from positive pair together and push away the points from negative pair. Triplet loss (Hoffer & Ailon, 2015) further requires a margin between the distance of the positive pair and the distance of the negative pair. One of the main issues of contrastive and triplet losses is that the number of possible pairs or triplets is extremely large for a large dataset.

A reasonable solution to address the sampling issue is mining samples that are the most informative for training, also known as “hard mining”. There is a large body of works addressing this problem (Schroff et al., 2015; Mishchuk et al., 2017; Harwood et al., 2017; Yuan et al., 2017; Wu et al., 2017). Semi-hard mining (Schroff et al., 2015) tries to find triplets in a training batch, for which the distance of the positive pair and the distance of the negative pair are within a certain margin. HardNet (Mishchuk et al., 2017) is designed to mine some of the hardest triplets within one training batch.

Designing structured losses to consider all the possible training pairs or triplets within one training batch and perform “soft” hard mining can be an alternative solution for hard mining (Song et al., 2017; 2016; Ustinova & Lempitsky, 2016). Lifted structured loss (Song et al., 2016) exploits all triplets in a training batch and provides a smooth loss function for hard mining. A few deep clustering based losses (Law et al., 2017; Song et al., 2017) have also been proposed to solve the problem. Proxy NCA (Movshovitz-Attias et al., 2017) proposes to learn semantic proxies for training data and use a NCA loss for training. Applying hard mining with proxies is more efficient than with samples.

In face verification, quite a few works have shown that training a classifier and using the output of the second last layer as embedding performs reasonably well (Wang et al., 2017b). NormFace (Wang et al., 2017a) and SphereFace (Liu et al., 2017) suggest to  $\ell_2$ -normalize both the embeddings and classifier weights. In order to achieve promising results, a learnable or fixed scalar is usually required to be multiplied to the final logits (Ranjan et al., 2017). There are some very preliminary discussions (Wang et al., 2017a; Ranjan et al., 2017) about how this scalar influences the final embedding.

This paper shows that the scalar can be seen as the temperature parameter of the softmax function in Hinton et al. (2015). We analyze how this temperature parameter controls the distribution, especially the compactness, of the embedding by assigning different gradients to different samples depending on their positions w.r.t. the boundary of the classifier. **Inspired by these findings, we propose a “heating-up” strategy for training the embedding, which uses increasing temperature while training the classifier.** The proposed method makes the embedding trained with the softmax function and the cross entropy loss achieve comparable or better performance than state-of-the-art deep metric learning methods.

## 3 REVISITING SOFTMAX EMBEDDING WITH TEMPERATURE

Given a set of  $n$  training samples with label  $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$ , where  $\mathbf{x}_i \in \mathbb{R}^d$  is the data sample,  $d$  being the number of dimension for the training data, and  $y_i \in \{1, \dots, M\}$  is the category label of sample  $\mathbf{x}_i$ ,  $M$  being the number of categories for training samples, we try to learn an embedding function  $\mathbf{f}(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^k$ , which maps a data sample to a vector in  $\mathbb{R}^k$ , such that for all  $i, j, p$  with  $y_i = y_j \neq y_p$ ,  $l(\mathbf{f}(\mathbf{x}_i), \mathbf{f}(\mathbf{x}_j)) < l(\mathbf{f}(\mathbf{x}_i), \mathbf{f}(\mathbf{x}_p))$ , where  $l(\cdot, \cdot) : \mathbb{R}^k \times \mathbb{R}^k \rightarrow \mathbb{R}$  is a distance function.

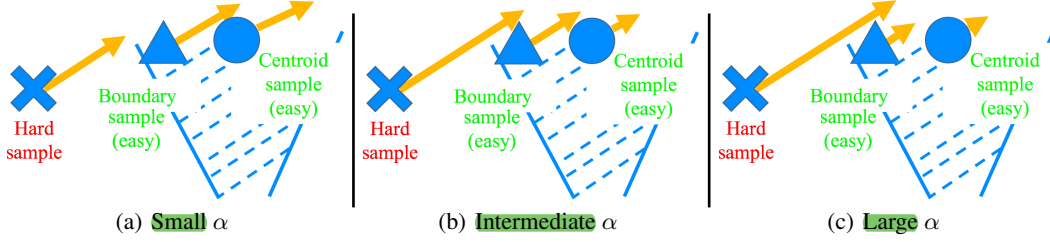


Figure 1: Different  $\alpha$  values control the final embeddings by assigning different gradients to different samples. Orange arrows show gradients.

We call  $\mathbf{f}(\mathbf{x}) \in \mathbb{R}^k$  the embedding of the data sample  $\mathbf{x}$  and use  $\mathbf{f}$  as  $\mathbf{f}(\mathbf{x})$  to simplify the notation. Considering training a linear classifier  $\mathbf{W} = [\mathbf{w}_1, \dots, \mathbf{w}_M] \in \mathbb{R}^{k \times M}$  and  $\mathbf{b} = [b_1, \dots, b_M]^T \in \mathbb{R}^M$ ,  $\mathbf{z} = [z_1, \dots, z_M]^T = \mathbf{W}^T \mathbf{f} + \mathbf{b} \in \mathbb{R}^M$  is called the logits. The probability that sample  $\mathbf{x}$  belongs to category  $m \in \{1, \dots, M\}$  can be predicted by the softmax function as:

$$p(m|\mathbf{x}) = \frac{\exp(z_m/T)}{\sum_{j=1}^M \exp(z_j/T)} = \frac{\exp(\alpha z_m)}{\sum_{j=1}^M \exp(\alpha z_j)} \quad (1)$$

$T$ , which is normally set to 1, is the temperature as mentioned in Hinton et al. (2015). We set  $\alpha = 1/T$  as the reciprocal of the temperature to simplify the notation in the paper.

Assuming the ground-truth distribution of the training sample is  $q(m|\mathbf{x})$ , generally  $q(m|\mathbf{x})$  is a Dirac delta function, which equals 1 if  $m = y$  and 0 otherwise, where  $y$  is the ground-truth label of  $\mathbf{x}$ , the cross entropy loss with respect to  $\mathbf{x}$ , and its gradient with respect to  $z_m$  are defined as:

$$\ell(\mathbf{x}, \alpha) = - \sum_{m=1}^M \log(p(m|\mathbf{x}, \alpha)) q(m|\mathbf{x}) \quad \text{and} \quad \frac{\partial \ell}{\partial z_m} = \alpha(p(m|\mathbf{x}, \alpha) - q(m|\mathbf{x})) \quad (2)$$

Considering  $z_m = \mathbf{w}_m^T \mathbf{f} + b_m$ , we have:

$$\frac{\partial \ell}{\partial \mathbf{f}} = \sum_{m=1}^M \frac{\partial \ell}{\partial z_m} \frac{\partial z_m}{\partial \mathbf{f}} = \alpha \sum_{m=1}^M (p(m|\mathbf{x}, \alpha) - q(m|\mathbf{x})) \mathbf{w}_m \quad (3)$$

The proposed ‘‘heating-up’’ idea is based on an observation that different  $\alpha$  values will assign gradients with different magnitudes to different samples and thus change the distribution of the final embeddings. To show this, since  $\alpha$ , the norm of the embedding and the norm of the classifier weights are all affecting the softmax function (Eq.(1) and (3)), we follow Wang et al. (2017a); Ranjan et al. (2017) and  $\ell_2$ -normalize the classifier weights  $\mathbf{w}_i$  and feature  $\mathbf{f}$  to have unit norm. The normalized features and weights are denoted as  $\hat{\mathbf{f}}$  and  $\hat{\mathbf{w}}_i$  respectively.

The following part of this section will first discuss how  $\alpha$  changes the gradient assignment (Sec. 3.1) and final embedding (Sec. 3.2) with  $\ell_2$ -normalized embeddings and weights, then the effect of normalization (Sec.3.3). Finally, we also discuss the embedding and gradient properties for an off-the-shelf classifier in 3.4.

### 3.1 GRADIENT ASSIGNMENT BY $\alpha$

In this subsection, we will show how training a deep classification network with different  $\alpha$  values affects the gradients of different training samples. From Eq. (1), when  $\alpha \rightarrow +\infty$ ,  $p(m|\mathbf{x}, \alpha)$  satisfies:

$$\lim_{\alpha \rightarrow +\infty} p(m|\mathbf{x}, \alpha) = \begin{cases} 1/K & z_m = \max(z_1, \dots, z_M) \\ 0 & \text{otherwise,} \end{cases} \quad (4)$$

where  $K$  is the number of logits whose value equals the maximum logits value. On the other hand, if  $\alpha$  approaches 0, the predicted probability will approach the uniform distribution. In other words, as  $\alpha$  increases, the predicted probability will become more ‘‘spiky’’ at the logits that have the largest value.

We define 2 types of training samples as in Fig. 1. In the figure, all data samples (crosses, triangles and circles) belong to the same category. All the samples which are in the area marked by blue

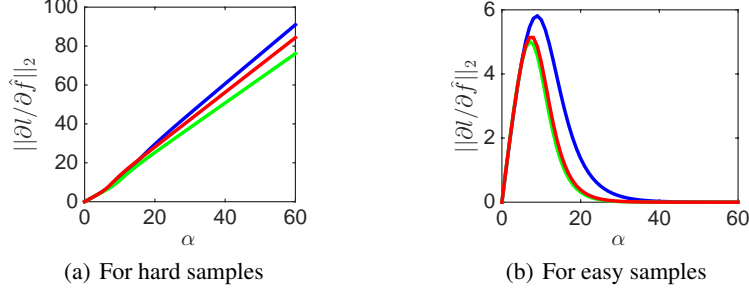


Figure 2: Relation between  $\alpha$  and the magnitude of the gradient with respect to embedding for both “hard” and “easy” samples.

**dashed lines** will be classified as the correct category by the classifier. The training samples can be divided into: (i) “Hard” samples (crosses): samples that are not classified as the correct category ( $\{\mathbf{x} : \exists m \neq y, z_m \geq z_y\}$ ); (ii) “Easy” samples: samples being correctly classified by the classifier ( $\{\mathbf{x} : \forall m \neq y, z_y > z_m\}$ ). There are two subtypes of samples in “Easy” category: “Boundary” samples (triangles) are samples close to the decision boundary; “Centroid” samples (circles) are samples laying close to the center of the region that belongs to the category.

The gradient  $\frac{\partial \ell}{\partial \mathbf{f}}$  of the loss w.r.t. the normalized embedding (i.e. Eq. (3) with  $\hat{\mathbf{f}}$  and  $\hat{\mathbf{w}}_i$  instead of  $\mathbf{f}$  and  $\mathbf{w}_i$ ) contains  $M$  terms in the sum, one term for each of the  $M$  categories. There is 3 types of terms: type 1, term with respect to the ground-truth category; type 2, term with respect to the logits that has the largest value and does not belong to the ground-truth category; type 3, other terms. We study the behavior of these terms for very small and very large  $\alpha$  for “hard” and “easy” samples. First of all, it’s easy to see that for  $\alpha \rightarrow 0$ , the magnitude of the gradient of any sample will approach 0.

For  $\alpha \rightarrow +\infty$ , considering “hard” samples first, the magnitude of the type 1 term ( $\alpha(p(y|\mathbf{x}, \alpha) - q(y|\mathbf{x}))\hat{\mathbf{w}}_y$ ) will approach  $+\infty$  as  $\alpha \rightarrow +\infty$ , because  $p(y|\mathbf{x}, \alpha)$  will approach either 0 or  $1/K$  ( $K \geq 2$ )<sup>1</sup> and  $q(y|\mathbf{x}) = 1$ . Similarly, the magnitude of the term of type 2 will also approach  $+\infty$ . For other terms, due to the property of the exponential function, that if  $z_m \neq \max(z_1, \dots, z_M)$ ,  $\lim_{\alpha \rightarrow +\infty} \alpha p(m|\mathbf{x}, \alpha) = 0$ , the magnitude of any term of type 3 will decrease to 0. Therefore, for “hard samples”, as  $\alpha \rightarrow +\infty$ , unless in some special cases<sup>2</sup>, the magnitude of the gradient with respect to the normalized embedding will approach infinity. Fig. 2(a) shows the magnitude of the gradient with respect to the embedding when  $\alpha$  changes for three different “hard” samples derived from the network in Sec.4.

Considering the term of type 1 for “easy” samples, since  $\lim_{\alpha \rightarrow +\infty} p(y|\mathbf{x}, \alpha) = 1$  and  $\lim_{\alpha \rightarrow +\infty} \alpha(p(y|\mathbf{x}, \alpha) - 1) = 0$ , the magnitude for this term will approach 0. For other terms, the magnitude will also approach 0. Therefore, the magnitude of the gradient will always approach 0. Fig. 2(b) shows how the magnitude of the gradient with respect to the embedding changes with  $\alpha$  for three different “easy” samples.

Overall, with large  $\alpha$ , the magnitudes of the gradients for hard samples will become very large, while the magnitudes of gradients for an easy samples will become very small (Fig. 1(c)). While, with small  $\alpha$ , the network will assign gradients of similar magnitudes to all the samples (Fig. 1(a)). Choosing an intermediate  $\alpha$  value is a trade-off (Fig. 1(b)). Gradient assignment for different samples will greatly affect the final embedding as discussed in the next section.

### 3.2 DISTRIBUTION OF FINAL EMBEDDING AND THE “HEATING-UP” STRATEGY

To illustrate the influence of  $\alpha$  on the embedding distribution, we show the embeddings obtained on the MNIST dataset in Fig. 3(a)-3(d), when using different  $\alpha$  values during training. Different colors represents different digits, and each diamond corresponds to the weight of the classifier of the corresponding digit. We slightly shift the weights towards the origin for better visualization. The base model is LeNet (LeCun et al., 2015), and the number of nodes in the second last layer is set to 2 for

<sup>1</sup>  $K$  can’t be 1, otherwise  $\mathbf{x}$  is an “easy” sample.

<sup>2</sup> For example, two terms having exactly the same magnitude but opposite direction.

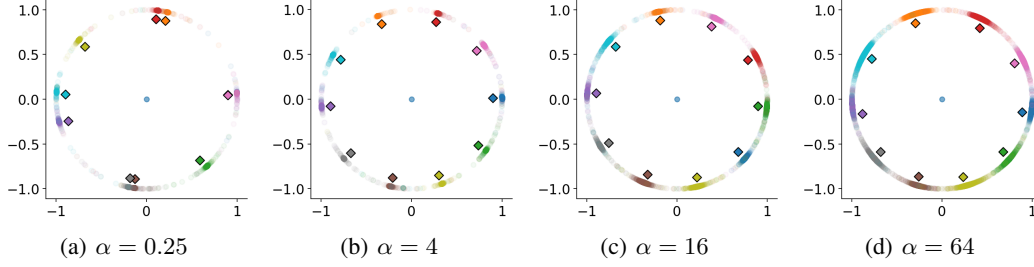


Figure 3: Embeddings trained with  $\ell_2$ -normalization and different  $\alpha$  values.

visualization. In the dataset, 50,000 samples are used for training, and 10,000 different test samples are used to draw the figure. All the features and classifier weights are  $\ell_2$ -normalized during training.

When training with small  $\alpha$  (i.e. **high temperature**), the network will assign **similar gradients** to all the samples (see Fig. 1(a)). Since the “hard” samples are more important to the classifier for improving the accuracy, updating “hard” samples and “easy” samples equally will make the training inefficient or even have difficulty to converge (Fig. 3(a)). However, choosing very large  $\alpha$  (i.e. very low temperature) for training will assign large gradients to “hard” samples and very small gradients to all “easy” samples ( “boundary” and “centroid”). Since the “boundary” samples will not get enough update, they will stay near the decision boundary. Therefore, the embedding of the samples of the same category will not be compact (Fig. 3(d)). As a consequence, a good trade-off is to use an intermediate temperature for training (see Fig. 1(b)), where “centroid” samples will get small gradients, “boundary” samples will get intermediate gradient, and “hard” samples will get large gradients. Comparing classifiers trained with different  $\alpha$  values, features of the same category are more compact for the model trained with smaller  $\alpha$  values (Fig. 3(d)  $\rightarrow$  3(b)).

We further propose a “heating-up” strategy, that consists in starting training with a low or medium temperature, such that “hard” samples will get large gradients to update and become “easy” samples soon. After that, the temperature will be increased and therefore “boundary” and “centroid” samples will also get enough gradients for updating. Therefore, the final embedding of the samples of the same category will become more compact than those of the model trained with the starting temperature only. Multiple strategies could be defined to increase the temperature during training. We tried: (i) gently increasing the temperature; (ii) training with the starting temperature until convergence and using a higher temperature to fine-tune the trained network. Both methods lead to similar performance. However, since the former method would introduce an additional parameter to control the speed of increase of the temperature, we used the latter in our experiments in Sec. 4.

### 3.3 INFLUENCE OF THE NORMALIZATION

We will here discuss the influence of the normalization, recall that we denote the original embedding as  $\mathbf{f}$  and the  $\ell_2$ -normalized embedding as  $\hat{\mathbf{f}}$ . The Jacobian matrix of  $\hat{\mathbf{f}}$  with respect to  $\mathbf{f}$  is:

$$\mathbf{J}_{\hat{\mathbf{f}}}(\mathbf{f}) = \frac{1}{\|\mathbf{f}\|_2}(\mathbf{I} - \hat{\mathbf{f}}\hat{\mathbf{f}}^T), \quad (5)$$

where  $\mathbf{I}$  is the identity matrix. Considering Eq. (3) and the chain rule, we have:

$$\frac{\partial \ell}{\partial \mathbf{f}} = \left(\frac{\partial \ell}{\partial \hat{\mathbf{f}}}\right)^T \mathbf{J}_{\hat{\mathbf{f}}}(\mathbf{f}) \quad (6)$$

Considering the norm  $\|\mathbf{f}\|_2$  in the denominator, the magnitude of the gradient is inversely proportional to the norm of the embedding. Therefore, even if the normalized embeddings are the same, the gradients w.r.t. the embeddings are still different for embeddings with different norms. The embedding with larger norm will have smaller gradient. This may seem as a problem, and one possible solution would be to remove the norm term  $\|\mathbf{f}\|_2$  in the denominator. We tried this idea for the experiments in Sec. 4, but it did not give significant improvement. The reason for that is likely that since  $\partial \ell / \partial \mathbf{f}$  and  $\mathbf{f}$  are always orthogonal (Wang et al., 2017a), updating the feature along the direction of gradient cannot change much the norm of the feature. We did observe that, when applying  $\ell_2$ -normalization to the feature during training, the norms of features before normalization are very similar. On the

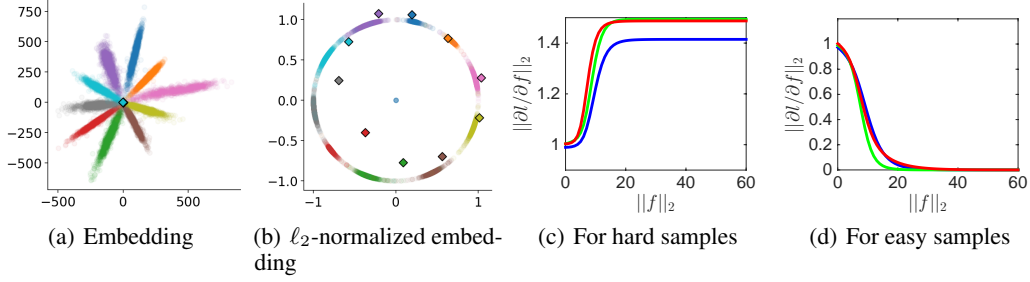


Figure 4: 4(a)-4(b): Embedding and normalized embedding of a classifier trained without normalization. 4(c)-4(d): Relation between the norm of the feature and the magnitude of the gradient with respect to feature for a classifier trained without normalization.

contrary, when training without normalization, the norm of the features may have large variations (see Sec. 3.4). For numerical stability and ease of implementation, we use Eq. (6) to calculate the gradient. Therefore, the gradient analysis in Sec. 3.1 and 3.2 still holds for unnormalized features before normalization.

We empirically found out that batch normalization (Ioffe & Szegedy, 2015) without the learned scale<sup>3</sup>,  $\hat{BN}(\cdot)$ , works slightly better than  $\ell_2$ -normalization. We propose to define the batch normalized embedding as:

$$\hat{\mathbf{f}}_{BN} = \hat{BN}(\mathbf{f})/\sqrt{k} \quad (7)$$

where  $k$  is the number of dimensions of  $\mathbf{f}$ . Batch normalization tries to make each dimension of the embedding have zero mean and unit variance. Therefore, after batch normalization, the norm of the embedding is roughly  $\sqrt{k}$ , and the normalized feature  $\hat{\mathbf{f}}_{BN}$  has norm close to 1, which is similar to  $\ell_2$  normalization. Batch normalization may work better than  $\ell_2$ -normalization because in fine-grained recognition problem, many embeddings from different categories can be very similar. Batch normalization removes the mean and rescales the embedding thus creating more variance. For classifier weights,  $\ell_2$ -normalization always gives us promising result.

### 3.4 COMPARING WITH OFF-THE-SHELF CLASSIFIER

It is interesting to compare the embedding of the proposed approach with the embedding from an off-the-shelf classifier (trained with unnormalized features and weights,  $\alpha = 1$ ) (see Fig. 4(a) for MNIST). As observed in other works (Wang et al., 2017a; Ranjan et al., 2017), (i) the magnitude of the embedding can be extremely large; (ii) the embedding is not “compact”. Even if the feature is  $\ell_2$ -normalized (Fig. 4(b)), the embedding is still not as “compact” as the feature trained with normalization and proper  $\alpha$  (Fig. 3(b)).

The norm of the embedding tends to be large because “easy sample” with larger norm will produce smaller loss (Wang et al., 2017a). To understand why the embeddings from the same category are not compact, examining the gradient is also the key. By setting  $\alpha = 1$  in Eq. (3), when  $\|\mathbf{f}\|_2 \rightarrow +\infty$ , for the “easy” sample, the magnitude of the gradient will approach 0. While for the “hard” sample, the magnitude will approach a constant, generally different from 0. Similarly to Fig. 2(a) and 2(b), the relation between the magnitude of the gradient w.r.t. the norm of the feature for some “hard” and “easy” samples is shown in Fig. 4(c) and 4(d). As boundary features trained with large  $\alpha$ , boundary features with large norm will not get enough update thus leading to a not compact embedding.

Training with  $\ell_2$ -normalization for the feature is different from simply applying normalization to the final feature in the test phase. As shown in Eq.(5) and (6),  $\ell_2$ -normalization will change the gradient of each sample during training.

<sup>3</sup>in this paper, batch normalization always refers to batch normalization without learned scale



## 4 EXPERIMENTS ON METRIC LEARNING

We conduct experiment on the following fine-grained datasets. using the training/test splits of Movshovitz-Attias et al. (2017). In all these datasets, the categories in the training and test splits do *not* overlap.

- **Cars** (Car196) (Krause et al., 2013) is a fine-grained car category dataset, which contains 16,185 images of 196 car models. 8,054 images of the first 98 categories are used for training, while the 8,131 images of the other 98 categories are used for test.
- **Caltech-UCSD Birds-200-2011** (CUB200) (Welinder et al., 2010) is a fine-grained bird category dataset which contains 11,788 images of 200 species of birds. 5,864 images of the first 100 species are used for training, while the 5,924 images of the other 100 species are used for test.
- **Stanford Online Product** (Product) dataset (Song et al., 2016) contains 120,053 images of 22,634 products categories. 59,551 images of 11,318 categories are used for training, while the other 60,502 images from 11,316 categories are kept for test.
- **In-shop Clothes Retrieval** (Fashion) dataset (Liu et al., 2016) contains 54,642 images of 11,735 categories of fine-grained clothes. The dataset is split into 3 subsets. 52,712 images of 7,982 categories are used for training. The other 28,760 images of 3985 categories are kept for test, split into a gallery set (12,612 images) and a query set (14,218 images).

### 4.1 IMPLEMENTATION DETAILS

We use the TensorFlow Deep Learning framework (Abadi et al., 2016) to implement the proposed method. For fair comparison, we exactly follow the details in Movshovitz-Attias et al. (2017). GoogLeNet V1 (Szegedy et al., 2015) from TensorFlow slim is used as the base network. The network is pre-trained with ILSVRC 2012-CLS data (Russakovsky et al., 2015). The input images are all resized to  $256 \times 256$ . For training, the resized images are randomly crop to  $224 \times 224$  with random horizontal flipping. In test phase, we only used one single center crop as in Movshovitz-Attias et al. (2017). For Car196, CUB200 and fashion datasets the network is fine-tuned by SGD optimizer with 0.9 momentum. The learning rate is set to 0.004. For product dataset, the optimizer is ADAM with learning rate of 0.01. The embedding size is set to 64 and the batch size is set to 32. We choose  $\alpha = 16$  for all the datasets, as an “intermediate” temperature, it works well for different embedding sizes (see Sec. 4.4). For “heating-up”,  $\alpha$  will decrease (temperature increases on the other hand) from 16 to 4 and the learning rate will decrease to 1/10 of the original learning rate. The training process usually converges within 50 training epochs, which is similar to the fastest state-of-the-art method, ProxyNCA (Movshovitz-Attias et al., 2017). Our implementation is available at: [https://github.com/ColumbiaDVMM/Heated\\_Up\\_Softmax\\_Embedding](https://github.com/ColumbiaDVMM/Heated_Up_Softmax_Embedding).

Table 1: NMI and Recall(%) for Car196 dataset

	[1]	[2]	[3]	[4]	[5]	[6]	SM	LN	BN	HLN	HBN
NMI	53.35	56.88	54.44	61.12	59.50	64.90	59.52	62.40	65.81	66.87	<b>68.10</b>
R@1	51.54	52.98	58.11	67.54	64.65	73.22	60.76	68.59	71.12	71.93	<b>74.70</b>
R@2	63.78	66.70	70.64	77.77	76.20	82.42	73.58	78.55	80.62	81.68	<b>83.90</b>
R@4	73.52	76.01	80.27	85.74	84.23	86.36	82.50	86.18	87.82	88.34	<b>89.77</b>

Table 2: NMI and Recall(%) for CUB200 Dataset

	[1]	[2]	[3]	[4]	[5]	[6]	SM	LN	BN	HLN	HBN
NMI	55.38	56.50	59.23	56.87	59.90	59.53	57.19	59.23	59.20	60.34	<b>60.75</b>
R@1	42.59	43.57	48.18	50.08	49.78	49.21	44.02	46.86	47.27	49.68	<b>50.68</b>
R@2	55.03	56.55	61.44	62.24	62.34	61.90	55.86	59.79	59.67	61.85	<b>62.58</b>
R@4	66.44	68.59	71.83	73.38	<b>74.05</b>	67.90	68.18	71.56	71.89	73.08	73.82

Table 3: NMI and Recall(%) for Stanford Product Dataset

	[1]	[2]	[3]	[4]	[6]	SM	LN	BN	HLN	HBN
NMI	89.46	88.65	89.48	88.70	<b>90.60</b>	88.66	90.11	90.45	90.39	<b>90.61</b>
R@1	66.67	62.46	67.02	64.52	<b>73.70</b>	63.94	69.51	71.19	70.36	72.04
R@10	82.39	80.81	83.65	82.53	N/A	80.07	84.69	85.89	85.41	<b>86.25</b>
R@100	91.85	91.93	93.23	92.35	N/A	90.28	92.97	93.75	93.70	<b>93.80</b>

Table 4: Recall(%) for In-shop Clothes Retrieval Dataset

	[7]	[8]	SM	LN	BN	HLN	HBN
R@1	53.0	62.1	78.6	79.6	80.7	80.5	<b>81.1</b>
R@10	73.0	84.9	93.7	94.2	<b>94.4</b>	94.2	94.2
R@20	76.0	91.2	95.4	96.0	<b>96.1</b>	<b>96.1</b>	95.9
R@30	77.0	92.3	96.3	96.8	<b>96.9</b>	96.7	<b>96.9</b>

## 4.2 EVALUATION

Following other existing works on metric learning, we evaluate the clustering quality and the retrieval performance on the images of the test set. Following Song et al. (2017), all the features are  $\ell_2$ -normalized before calculating the evaluation metric. The normalized features performs slightly better than the unnormalized feature.

For clustering, the K-Means algorithm is run on all the embeddings of the test samples. The number of cluster is chosen to be the number of categories in the test set. Each test sample will be assigned a cluster index according to which cluster it belongs to. Normalized Mutual Information (NMI) (Schütze et al., 2008) between the clustering index and the ground-truth label is used as the metric for clustering. Note that NMI is invariant to the label permutation.

For retrieval, the performance is evaluated by Recall@K, which is also a widely used metric for this problem. Given a query sample from the test set, K samples from the rest of the test set (or the gallery set for the fashion dataset) with the smallest distance are retrieved. If any retrieved samples is from the same category as the query sample, the recall for this sample is set to 1, otherwise, 0. The reported Recall@K is the average recall on the whole test set.

We train a classifier on the training dataset with the softmax function and cross-entropy as baseline (SM). For the baseline classifier, in training, the features and the weights are not normalized and  $\alpha$  is set to 1. 4 different versions of classifiers trained with the proposed methods are used for evaluation:

- **LN**: softmax with  $\ell_2$ -normalized embedding,  $\ell_2$ -normalized weights and  $\alpha = 16$ .
- **BN**: softmax with batch normalized embedding,  $\ell_2$ -normalized weights and  $\alpha = 16$ .
- **HLN**: Heated-up model using  $\alpha = 4$  to fine-tune LN.
- **HBN**: Heated-up model using  $\alpha = 4$  to fine-tune BN.

We also compare the proposed method with multiple state-of-the-art metric learning methods. Existing literatures are using different base networks and different evaluation protocols. For fair comparison, only the methods using GoogleNetV1 as base network and Euclidean distance as the final evaluation metric are listed: [1] Triplet learning with semi-hard negative mining (Schroff et al., 2015), [2] Lifted structured loss (Song et al., 2016), [3] Learnable Structured Clustering (Song et al., 2017); [4] Deep Clustering Learning without spectral learning (Law et al., 2017); [5] Deep Metric Learning with Smart Mining (Harwood et al., 2017) and [6] ProxyNCA (Movshovitz-Attias et al., 2017). For the

Table 5: R@1(%) for Car196 with Different  $\alpha$  Values and Embedding Sizes

$\backslash$ MODEL	SM	$\alpha = 4$	$\alpha = 8$	$\alpha = 16$	$\alpha = 32$	$\alpha = 64$	HBN (16 $\rightarrow$ 4)
#DIM							
64	60.8	67.4	68.7	71.1	69.5	62.5	<b>74.0</b>
128	65.2	71.6	71.0	74.2	73.0	66.6	<b>77.5</b>
256	67.3	72.2	69.7	78.0	75.2	70.1	<b>80.1</b>



Fashion dataset, we compared with [7] FashionNet (Liu et al., 2016) and [8] Hard-Aware Deeply Cascaded Embedding (Yuan et al., 2017).

#### 4.3 METRIC LEARNING

The performances of all the methods on all 4 datasets are listed in Tables 1, 2, 3 and 4 respectively. The softmax baseline already shows comparable result with many other triplet loss based methods. The embeddings trained with either  $\ell_2$ -normalization or batch normalization improve the performance of the softmax baseline. Since in test phase, all the features are  $\ell_2$  normalized before calculating the metric, the performance gain is not coming from a simple  $\ell_2$  normalization to the final features. Batch normalization works slightly better than  $\ell_2$ -normalization. The “heated-up” models (HLN and HBN) show better performance in almost all the metrics compared to the embedding trained with a fixed temperature.

#### 4.4 EMBEDDING SIZE AND $\alpha$

We further study how different embedding sizes and  $\alpha$  values affect the retrieval performance. The size of the embedding is chosen in [64, 128, 256], and the  $\alpha$  value in [4.0, 8.0, 16.0, 32.0, 64.0]. The R@1 metric on the test set with different embedding sizes and  $\alpha$  values is reported in Tab. 5. Performances of the feature learned by softmax function without normalization and feature learned from “heated-up” model are also given. The “heated-up” model outperforms all the other models by a significant margin in all cases. Between models trained with fixed  $\alpha$  values, the model trained with  $\alpha = 16$  outperforms others.

### 5 DISCUSSION

We have discussed how the temperature parameter in the softmax function affects the distribution of the embedding in the second last layer of a deep classification model. Training with an intermediate temperature will lead to an intra-category compact and inter-category “spread-out” embedding which is beneficial for both clustering and retrieval. A “heating-up” method is also proposed to further improve the clustering and retrieval performance of the embedding by fine-tuning with a higher temperature. Our classifier based approach achieves good performance in metric learning problems with a simpler and more efficient training process than state-of-the-art methods.

#### ACKNOWLEDGMENTS

This material is based upon work supported by the United States Air Force Research Laboratory (AFRL) and the Defense Advanced Research Projects Agency (DARPA) under Contract No. FA8750-16-C-0166. Any opinions, findings and conclusions or recommendations expressed in this material are solely the responsibility of the authors and does not necessarily represent the official views of AFRL, DARPA, or the U.S. Government.

#### REFERENCES

- Marín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A System for Large-Scale Machine Learning. *OSDI*, 2016.
- Artem Babenko and Victor Lempitsky. Aggregating Deep Convolutional Features for Image Retrieval. In *ICCV 2015*, 2015.
- S. Chopra, R. Hadsell, and Y. LeCun. Learning a Similarity Metric Discriminatively, with Application to Face Verification. In *CVPR 2005*, 2005.
- Matthieu Guillaumin, Jakob Verbeek, and Cordelia Schmid. Is that you? metric learning approaches for face identification. In *CVPR 2009*, 2009.
- Ben Harwood, Vijay Kumar B G, Gustavo Carneiro, Ian Reid, and Tom Drummond. Smart Mining for Deep Metric Learning. In *ICCV 2017*, 2017.
- Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network. In *NIPS Workshop 2015*, 2015.

- Elad Hoffer and Nir Ailon. Deep Metric Learning Using Triplet Network. In *International Workshop on Similarity-Based Pattern Recognition, 2015*. Springer, 2015.
- Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML 2015*, 2015.
- Martin Koestinger, Martin Hirzer, Paul Wohlhart, Peter M Roth, and Horst Bischof. Large scale metric learning from equivalence constraints. In *CVPR 2012*, 2012.
- Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3D Object Representations for Fine-Grained Categorization. In *4th International IEEE Workshop on 3D Representation and Recognition, 2013*, 2013.
- Marc T. Law, Raquel Urtasun, and Richard S. Zemel. Deep Spectral Clustering Learning. In *ICML 2017*, 2017.
- Yann LeCun et al. Lenet-5, Convolutional Neural Networks. *url: <http://yann.lecun.com/exdb/lenet>*, 2015.
- Jung-Eun Lee, Rong Jin, and Anil K Jain. Rank-based distance metric learning: An application to image retrieval. In *CVPR 2008*, 2008.
- Giuseppe Lisanti, Svebor Karaman, and Iacopo Masi. Multichannel-kernel canonical correlation analysis for cross-view person reidentification. *ACM TOMM*, 2017.
- Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. SphereFace: Deep Hypersphere Embedding for Face Recognition. In *CVPR 2017*, 2017.
- Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations. In *CVPR 2016*, 2016.
- Anastasiya Mishchuk, Dmytro Mishkin, Filip Radenovic, and Jiri Matas. Working Hard to Know Your Neighbor’s Margins: Local Descriptor Learning Loss. In *NIPS 2017*, 2017.
- Yair Movshovitz-Attias, Alexander Toshev, Thomas K. Leung, Sergey Ioffe, and Saurabh Singh. No Fuss Distance Metric Learning Using Proxies. In *ICCV 2017*, 2017.
- Rajeev Ranjan, Carlos D. Castillo, and Rama Chellappa. L2-constrained Softmax Loss for Discriminative Face Verification. *arXiv:1703.09507 [cs]*, 2017.
- Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-Shelf: an Astounding Baseline for Recognition. In *CVPRW 2014*, 2014.
- Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015.
- Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A Unified Embedding for Face Recognition and Clustering. In *CVPR 2015*, 2015.
- Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. Introduction to Information Retrieval. *Cambridge University Press*, 39, 2008.
- Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep Metric Learning via Lifted Structured Feature Embedding. In *CVPR 2016*, 2016.
- Hyun Oh Song, Stefanie Jegelka, Vivek Rathod, and Kevin Murphy. Deep Metric Learning via Facility Location. In *CVPR 2017*, 2017.
- Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, et al. Going Deeper With Convolutions. In *CVPR 2015*, 2015.
- Evgeniya Ustinova and Victor Lempitsky. Learning Deep Embeddings with Histogram Loss. In *NIPS 2016*, 2016.

- Feng Wang, Xiang Xiang, Jian Cheng, and Alan L. Yuille. NormFace: L2 Hypersphere Embedding for Face Verification. In *ACM MM 2017*, 2017a.
- Jian Wang, Feng Zhou, Shilei Wen, Xiao Liu, and Yuanqing Lin. Deep Metric Learning with Angular Loss. In *ICCV 2017*, 2017b.
- P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical report, 2010.
- Chao-Yuan Wu, R. Manmatha, Alexander J. Smola, and Philipp Krähenbühl. Sampling Matters in Deep Embedding Learning. In *ICCV 2017*, 2017.
- Eric P Xing, Michael I Jordan, Stuart J Russell, and Andrew Y Ng. Distance metric learning with application to clustering with side-information. In *NIPS 2003*, 2003.
- Jieping Ye, Zheng Zhao, and Huan Liu. Adaptive distance metric learning for clustering. In *CVPR 2007*, 2016.
- Yuhui Yuan, Kuiyuan Yang, and Chao Zhang. Hard-Aware Deeply Cascaded Embedding. In *ICCV 2017*, 2017.
- Xu Zhang, Felix X. Yu, Sanjiv Kumar, and Shih-Fu Chang. Learning Spread-Out Local Feature Descriptors. In *ICCV 2017*, 2017.