Making Classification Competitive for Deep Metric Learning

Andrew Zhai
Pinterest, Inc.
andrew@pinterest.com

Hao-Yu Wu
Pinterest, Inc.
rexwu@pinterest.com

Abstract

Deep metric learning aims to learn a function mapping image pixels to embedding feature vectors that model the similarity between images. The majority of current approaches are non-parametric, learning the metric space directly through the supervision of similar (pairs) or relatively similar (triplets) sets of images. A difficult challenge for training these approaches is mining informative samples of images as the metric space is learned with only the local context present within a single mini-batch. Alternative approaches use parametric metric learning to eliminate the need for sampling through supervision of images to proxies. Although this simplifies optimization, such proxy-based approaches have lagged behind in performance. In this work, we demonstrate that a standard classification network can be transformed into a variant of proxy-based metric learning that is competitive against non-parametric approaches across a wide variety of image retrieval tasks. We address key challenges in proxy-based metric learning such as performance under extreme classification and describe techniques to stabilize and learn higher dimensional embeddings. We evaluate our approach on the CAR-196, CUB-200-2011, Stanford Online Product, and In-Shop datasets for image retrieval and clustering. Finally, we show that our softmax classification approach can learn high-dimensional binary embeddings that achieve new state-of-the-art performance on all datasets evaluated with a memory footprint that is the same or smaller than competing approaches.

1. Introduction

Learning image representations, also known as image embeddings, is a core problem of a variety of applications including face recognition [19], fine-grained retrieval [22] [28] [20], clustering [31], and visual search [32] [15] [33] [9]. Standard deep neural network metric learning approaches learn image representations through the local relationships between images in the form of *pairs* [3] [1] or *triplets* [8] [19]. Similarity-style supervision are used to train the representation such that similar images are close in embedding space and dissimilar images apart.

A core challenge with metric learning is sampling informative samples for training. As described in [19] [7] [28], negatives that are too hard can destabilize training, while negatives that are too easy result in triplets that have near zero loss leading to little contribution to network learning. The widely adopted Semi-hard sampling[19] technique provides a good balanced sampling strategy. Recent methods such as [22] [20] [7] [28] focus on addressing this sampling problem, many of which utilize the relationships of all images within the *batch* to form informative samples. These methods typically require a large batch size so that informative samples can be selected within the batch. In practice, batch size is constrained by the hardware memory size. Therefore, as the dataset size grows, one still faces the challenge of the diminishing probability of a randomly sampled batch containing any informative samples.

Another challenge with metric learning is difficulties in optimization. Since each optimization step only depends on local context within each minibatch, the training often takes a long time to converge [17] or converges to local a optima [25]. In contrast to triplet approaches, the proxy-based approach [17] approximates each semantic class using a proxy, and uses all proxies to provide global context for each training iteration. Such a training setup is similar to the standard classification task, and eliminates the challenge of sampling informative samples. Though proxy-based metric learning are attractive due to simplify training by removing sampling, they have scalability limitations similar to *extreme classification* [22] and have been outperformed by recent advancements in metric learning [28] [25] [18].

The goal of this paper is to investigate the effectiveness of softmax classification for metric learning. Our major contributions are as follows: 1) We empirically demonstrate that embeddings trained using a classification-based approach can achieve state of the art performance on public benchmarks while avoiding many of the issues that make training embeddings using metric-based approaches difficult. 2) We examine the connection between standard classification using softmax cross-entropy loss and proxy-based

Neighborhood Component Analysis (NCA) loss. A theoretical justification is provided to show that by removing the bias term in the final linear layer and L2 normalizing the inputs and weights, the classification task is a variant of metric learning. 3) We demonstrate that adding temperature scaling of the distances and Layer Normalization [14] improves the numerical stability of optimization, and that such simple modifications provide a strong baseline for metric learning. 4) We describe how to learn high-dimensional binary embeddings trained using our approach to achieve state-of-the-art retrieval performance with the same memory footprint as 64 dimensional float embeddings. 5) We address the *extreme classification* scalability limitations of proxy-based approaches through experiments demonstrating the effects of subsampling target classes for each training iteration. To the best of our knowledge, we are the first to show the effect of subsampling on proxy-based approaches empirically.

2. Related Works

Metric Learning Losses Metric learning approaches aim to learn a good embedding space such that the similarity between samples are preserved as distance between embedding vectors of the samples. The metric learning losses, such as contrastive loss[3] and triplet loss[8], are formulated to minimize intra-class distances and maximize inter-class distances. Recent approaches in metric learning design the loss function to consider the relationships of all the samples within the training batch[22][20][13][28][24], and achieve state-of-the-art performance in image retrieval datasets[26][16][22].

Training Sampling Sampling informative training samples plays an important role in metric learning as also suggested in [19][28]. Semi-hard sampling in conjunction with triplet-loss [19] has been widely adopted for many tasks. Distanced-weighted sampling [28] suggests that with a balanced mix of difficulties during training time, the image retrieval performance can be further improved. Hierarchical Triplet Loss [25] proposed that by merging similar classes dynamically during training time into a hierarchical tree, more informative samples can be drawn from such a structure and the loss also provides global context for training.

Ensembling Ensembling embeddings has been the most recent focus to further improve image retrieval performance. The ensembled embeddings are trained via boosting [18] or attending diverse spatial locations [27]. However, such ensembled embeddings trade off image retrieval performance with higher dimensions

Global Context via Active Memory Proxy loss [17] uses a small number of proxies in proxy bank to represent all positives and negatives samples in training data. Scalable NCA loss [29] further extends the idea to store all image embeddings in memory and perform approximate update for the stored embeddings.

3. Classification as Metric Learning

In this section, we will prove that **removing bias** and **L2-normalizing** inputs and weights before the final softmax cross-entropy layer make classification a variant of proxy-based metric learning. We first examine the Neighborhood Component Analysis (NCA) Loss used as the surrogate loss for ranking in [17]. With the set of dissimilar images denoted as Z and distance between embeddings of images x, y as $d(x, y)$:

$$L_{\text{NCA}}(x, y, Z) = -\log\left(\frac{\exp(-d(x, y))}{\sum_{z\in Z}\exp(-d(x, z))}\right) \quad (1)$$

We argue that by excluding the positive term $exp(-d(x, y))$, the proxy-NCA loss does not have the probabilistic meaning as proposed in original NCA paper[5]. The softmax cross-entropy loss (softmax loss) used in standard classification task, on the other hand, which is more similar to NCA loss in spirit, also tries to make x closer to y than any $z \in Z$.

$$L_{\text{cls}}(x, y, Z) = -\log\left(\frac{\exp(-d(x, y))}{\exp(-d(x, y)) + \sum_{z\in Z}\exp(-d(x, z))}\right) \quad (2)$$

With the static proxy assignment and the choice of distance function being the **cosine distance**:

$$d(x, y) = 1 - x^T y \quad (3)$$

The proxy-classification loss can then be expressed as standard softmax loss when input and weights are L2-normed:

$$\begin{aligned} L_{\text{cls}}(x, p_y, p_Z) &= -\log\left(\frac{\exp(-(1 - x^T p_y))}{\sum_{p_z\in(p_Z\cup p_y)}\exp(-(1 - x^T p_z))}\right) \\ &= -\log\left(\frac{\exp(x^T p_y))}{\sum_{p_z\in(p_Z\cup p_y)}\exp(x^T p_z))}\right) \end{aligned} \quad (4)$$

By using the network's penultimate layer's normalized output as the embedding, and the normalized weights of the last fully-connected layer as the proxies, we show that standard classifier training **without bias** in the last layer is just a variant of proxy-based metric learning. We refer this modified softmax loss as **normalized softmax loss**. This explains the strong baseline observed by simply using the activations of the network pre-trained on a classification task such as ImageNet[4]. We can also leverage the vast body of literature on the tricks for optimizing classification task training and performance.

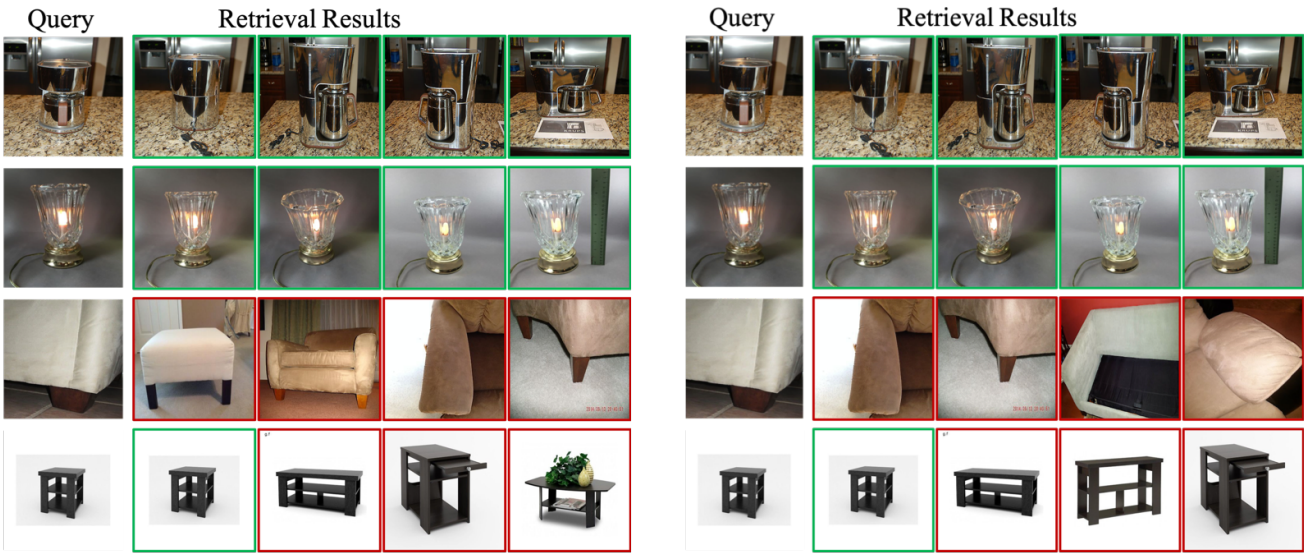

Figure 1. Retrieval results from the Stanford Online Products (SOP) dataset using our best performing embeddings in Table 6. (Left) Binary embedding results. (Right) Float embedding results.

4. Model

Based on the derivation in Section 3, it is very simple to convert any classification network to perform proxy-based metric learning. One only needs to remove the bias term in the last linear layer and add an L2 normalization module to the inputs and weights before softmax loss. To achieve state-of-the-art metric learning performing using softmax, we introduce the following components without adding additional parameters:

4.1. Temperature Scaling

As shown in [28], in high-dimensional spaces, the distance between two random points on the unit-sphere approaches normal distribution $N(\sqrt{2}, \frac{1}{2dim})$ where dim is the dimensionality. Such phenomena is also known as the *contrast-loss*[2] in high dimensional space. Therefore, to make the network focus on smaller differences in distance as the dimensionality increases, we add the temperature term σ to scale cosine distance before softmax loss:

$$L_{\text{cls}}(x, p_y, p_Z, \sigma) = -\log\left(\frac{\exp(\frac{x^T p_y}{\sigma}))}{\sum_{p_z \in (p_Z \cup p_y)} \exp(\frac{x^T p_z}{\sigma}))}\right) \tag{5}$$

4.2. Layer Normalization

The layer normalization without affine parameters[14] is added immediately after the final pooling layer of the feature model (e.g. GoogleNet [23]'s pool5 layer) to normalize the feature dimension of our embeddings to have a distribution of values centered at zero. This allows us to easily binarize embeddings via thresholding at zero. We also show empirically through ablation experiments in Section 6.4.2 that layer normalization helps the network better initialize new parameters and reach better optima.

4.3. Subsampling

Fitting all proxies in memory during training may not be possible with high-dimensional embeddings and a large number of classes. Furthermore, even if we could fit a large number of classes in memory, having softmax training scale linearly in complexity with the number of classes is an undesirable property for efficient training.

To address this issue, we propose a **subsampling module**. For each training batch, only the correct classes and a randomly sampled subset of the wrong classes are used for optimization. We show empirically that subsampling slightly degrades the performance, but we can achieve competitive performance with only a 10% subsampling ratio of the classes on the Stanford Product dataset in Section 6.5.

Although subsampling is not necessary for the existing datasets, the simple subsampling module alleviates the constraint of the number of classes when training the classifier. It also offers the flexibility to trade-off the dimension of the embedding and the ratio of subsampling.

5. Datasets

We evaluate our method on commonly used image retrieval tasks with the standard train/test split protocols. Specifically, we use the following datasets:

CARS-196 [16]. 16,185 images distributed over 196 classes of cars. The first 98 classes with 8,054 images are used for training and the other 98 classes with 8,131 images

are used for testing. The test set is both the query and index set.

CUB-200-2011 [26]. 11,788 images distributed over 200 species of birds. The first 100 classes with 5,864 images are used for training and the other 100 classes with 5,924 images are used for testing. The test set is both the query and index set.

Stanford Online Products (SOP) [22]. 120,053 images distributed over 22,634 classes of products. 11,318 classes with 59,551 images are used for training and the other 11,316 classes with 60,502 images are used for testing. The test set is both the query and index set.

In-shop Clothes Retrieval [34]. 54,642 images distributed over 11,735 classes of clothing items. Following standard protocol, only 7,982 classes with 52,712 images are used for training and testing. 3,997 classes with 25,882 images are used for training and 3,985 classes with 28,760 images are used for testing. The test set is split into a 14,218 image query set and a 12,612 image index set, both containing examples from each of the 3,985 classes.

6. Experiments

Following the same evaluation protocol as in [22], we compare our method using Recall@K to measure retrieval quality and NMI to measure clustering quality. To compute Recall@K, during testing we first compute the embeddings for all test images. For each test image, we then retrieve the top K images from the test set, excluding the query image itself, using cosine similarity. A recall score of 1 is assigned to every query image that has at least one image that matches the query's label from the K retrieved images, 0 otherwise. Recall@K is then the average of the recall scores for all queries.

We first investigate how embeddings trained with softmax classification compare against embeddings trained with existing metric learning losses using the same featurizer and embedding dimension. Following our loss comparisons, we conduct ablation studies on different design choices of our approach on CUB-200-2011 [26] (Section 6.4). One facet of interest we study in detail is the dimensionality of our embeddings (Section 6.3) and its relation to performance. Next, we investigate how softmax embeddings are affected by class subsampling (Section 6.5), addressing the key scalability concern of softmax loss where training complexity is linear with the number of classes. Finally in Section 6.6, we show that our method outperforms state-of-the-art methods on several retrieval datasets.

6.1. Implementation

All experiments were executed using PyTorch and a Tesla V100 graphic card. We compare our softmax experiments against common architectures used in metric learning including GoogleNet [23], GoogleNet with Batch Normalization [11], and ResNet50 [6]. We initialize our networks with pre-trained ImageNet ILSVRC-2012 [4] weights, with GoogleNet weights copied from the Caffe [12] trained model as in [22] [13] [20]. We add a randomly initialized fully connected layer to the pool5 features of each architecture to learn embeddings of varying dimensionality. To simplify the sensitivity to the initialization of the fully connected layer, we add a Layer Normalization [14] without additional parameters between the pool5 and fully connected layer (See Section 6.4.2 for the ablation study). We L2 normalize both the embedding and class weights before Softmax and use a temperature of 0.05 for all experiments (See Section 6.4.3 for the temperature ablation study).

Unless otherwise stated, we first train for 1 epoch, updating only new parameters for better initialization. We then optimize all parameters for 30 epochs with a batch size of 75 with the same learning rate. We construct our batch by sampling 25 examples per class for all datasets except the Stanford Online Products where we sample 5 due to few examples per class in the dataset (See Section 6.4.1 for the ablation study). We alternate between base learning rates of 1e-2 and 1e-3 depending on both dataset and model, unsurprising given the optimization variations of our different architectures (e.g. ResNet50 has Batch Normalization [11] which allows higher learning rates to be used unlike GoogleNet). We use SGD with momentum of 0.9, weight decay of 1e-4, and gamma of 0.1 which we apply to reduce the learning rate at epoch 15. During training, we apply horizontal mirroring and random crops from 256x256 images; during testing we only center crop from the 256x256 image. Following [13] [17], we crop to 227x227 for GoogleNet and 224x224 for the rest.

6.2. Loss Function Comparisons

We compare our normalized softmax loss against existing metric learning losses. To focus on contributions from the loss functions, we leave comparisons against methods that ensemble models [30], modify the feature extractor architecture [27], or propose complex activation paths between the featurizer and final embedding [18] for Section 6.6.

We present Recall@K and NMI results on three standard retrieval datasets in Table 1, Table 2, and Table 3, comparing against reported performance of methods trained with model architectures of GoogleNet, GoogleNet with Batch Normalization (BNInception), and ResNet50 respectively. For GoogleNet with Stanford Online Products only, we saw around a 1% Recall@1 improvement by training all parameters from start with 10x learning rate on new parameters when compared with models trained with our standard finetuning procedure.

As shown, our approach compares very strongly against

existing baselines, achieving the best Recall@K and NMI performance for many of the tasks. When fixing dimensionality to 512, we see that the performance improvements of our softmax embeddings across architectures mirror classification performance on ImageNet ILSVRC-2012. We hope our results help disentangle performance improvements of existing metric learning methods due to advancements in methodology versus changes of base feature models.

6.3. Embedding Dimensionality

To study the effects of dimensionality on our softmax embeddings, we keep all optimization parameters fixed with the exception of the dimensionality of the added fully connected layer. We have consistently observed that dimensionality is directly related to retrieval performance for our Softmax trained embeddings. Two examples of this across different datasets (CARS-196 and CUB-200-2011) and model architectures (ResNet50 and GoogleNet) are shown in Figure 2. Interestingly, this is in contrast to reported behaviors for previous non-parametric metric learning methods [13] [22] [21], showing that dimensionality does not significantly affect retrieval performance. This difference is seen clearly When comparing R@1 across dimensionality for CUB-200-2011 with GoogleNet in Figure 2 with the same dataset and model combination from [22].

Higher dimensional embeddings lead to an increase in retrieval performance. Lower dimensional embeddings however are preferred for scalability to reduce storage and distance computation costs especially in large scale applications such as visual search [15]. We observe however that as we increase dimensionality of our softmax embeddings, the optimizer does not fully utilize the higher dimensional metric space. Instead, we see that feature dimensions start relying less on the magnitude of each feature dimension and instead rely on the sign value. In Figure 2, we see for both datasets that the Recall@1 performance of binary features (thresholding the feature value at zero) converges with the performance of the float embeddings. This is a consistent result we see across datasets and model architectures. We show that training high dimensional embeddings and binarizing leads to the best trade-off of performance and scalability as described in Section 6.6.

6.4. Ablation Studies

In this set of experiments we report the effect on Recall@1 with different design choices in our approach on CUB-200-2011. We train the ResNet50 with embedding dimension of 512 variant as in Table 3. We fix all hyperparameters besides the one of interest.

6.4.1 Class Balancing

As seen in Table 4, class balancing seems beneficial over random sequential iteration over the dataset. Given the fine-grained nature of the CUB-200-2011 dataset, we hypothesize that by having enough examples of a particular class in one mini-batch, the gradient signal learned can better separate a particular class against other fine-grained negative class proxies. When too few distinct classes exist in the minibatch however, the bias of separating few distinct classes may introduce noise to the optimization, resulting in lowered performance.

6.4.2 Layer Normalization

We utilize Layer Normalization [14] without parameters to standardize activation values after pooling to help the initialization of our training. With 100 classes in CUB-200-2011, we expect that a random classifier would have a loss value of roughly $-\ln(\frac{1}{100}) = 4.6$. As shown in Table 3, this occurs when training with Layer Normalization, but not without. We have found incorporating Layer Normalization in our training allows us to be robust against poor weight initialization of new parameters across model architectures, alleviating the need for careful weight initialization.

6.4.3 Temperature

We see in Table 5 the effects of varying temperature. As described in Section 4.1, due to distances becoming less distinguishable with higher dimensionality, temperature is necessary to improve stability of training. This can be seen as temperature values close to one have sharp drops in performance. Similarly temperature values too low also result in low performance. We hypothesize the scaling from low temperatures result in augmenting too much noise as the variance of seen distances increases. The robustness of our temperature setting of 0.05 however is validated as we fix this setting across all experiments we run, regardless of dimensionality of embedding learned.

6.5. Subsampling for Classification Scalability

We apply the subsampling methodology described in Section 4.3 to the Stanford Online Products dataset, the retrieval dataset with the largest number of classes (Section 5), using ResNet50 to train a 2048 dimensional embedding. We present our findings in Figure 6.5, showing that with only 10% of the classes available during the forward pass of training, we can reach a R@1 performance comparable to using all classes (1% drop in performance). When using 1% of classes, we reach a R@1 of 75.7 (a better performance than most methods in Table 6). When using 0.1% of classes, we reach a R@1 of 72.0. As we can see,

	SOP					CARS-196					CUB-200				
Recall@K	1	10	100	1K	NMI	1	2	4	8	NMI	1	2	4	8	NMI
Contras.^{128} [22]	42.0	58.2	73.8	89.1	-	21.7	32.3	46.1	58.9	-	26.4	37.7	49.8	62.3	-
Lift. Struc^{128} [22]	-	-	-	-	-	49.0	60.3	72.1	81.5	55.0	47.2	58.9	70.2	80.2	55.6
Lift. Struc^{512} [22]	62.1	79.8	91.3	97.4	-	-	-	-	-	-	-	-	-	-	-
Hist Loss^{512} [24]	63.9	81.7	92.2	97.7	-	-	-	-	-	-	50.3	61.9	72.6	82.4	-
Bin. Dev^{512} [24]	65.5	82.3	92.3	97.6	-	-	-	-	-	-	52.8	64.4	74.7	83.9	-
Npairs^{512} [20]	67.7	83.8	93.0	97.8	88.1	-	-	-	-	-	-	-	-	-	-
Npairs^{64} [20]	-	-	-	-	-	71.1	79.7	86.5	91.6	64.0	51.0	63.3	74.3	83.2	60.4
Angular^{512} [13]	**70.9**	**85.0**	**93.5**	**98.0**	**88.6**	71.4	81.4	87.5	92.1	63.2	54.7	66.3	76.0	83.9	61.1
NormSoftmax^{512}	69.0	84.5	93.1	97.8	88.2	**75.2**	**84.7**	**90.4**	**94.2**	**64.5**	**55.3**	**67.0**	**77.6**	**85.4**	**62.8**

Table 1. Recall@K and NMI across standard retrieval tasks. All methods are trained using GoogleNet for a fair comparison.

	SOP					CARS-196					CUB-200				
Recall@K	1	10	100	1K	NMI	1	2	4	8	NMI	1	2	4	8	NMI
Clustering^{64} [21]	67.0	83.7	93.2	-	89.5	58.1	70.6	80.3	87.8	59.0	48.2	61.4	71.8	81.9	59.2
Proxy NCA^{64}[17]	73.7	-	-	-	90.6	73.2	82.4	86.4	88.7	64.9	49.2	61.9	67.9	72.4	59.5
HTL^{512} [25]	**74.8**	**88.3**	94.8	**98.4**	-	81.4	88.0	92.7	95.7	-	57.1	68.8	78.7	86.5	-
NormSoftmax^{512}	73.8	88.1	**95.0**	98.3	89.8	**81.7**	**88.9**	**93.4**	**96.0**	**70.5**	**59.6**	**72.0**	**81.2**	**88.4**	**66.2**

Table 2. Recall@K and NMI across standard retrieval tasks. All methods are trained using BNInception for a fair comparison

	SOP					CARS-196					CUB-200				
Recall@K	1	10	100	1K	NMI	1	2	4	8	NMI	1	2	4	8	NMI
Margin^{128} [28]	72.7	86.2	93.8	98.0	90.7	79.6	86.5	91.9	95.1	69.1	**63.6**	**74.4**	83.1	**90.0**	69.0
NormSoftmax^{128}	75.2	88.7	95.2	98.4	90.4	81.6	88.7	93.4	96.3	72.9	56.5	69.6	79.9	87.6	66.9
NormSoftmax^{512}	**78.2**	**90.6**	**96.2**	**98.8**	**91.0**	**84.2**	**90.4**	**94.4**	**96.9**	**74.0**	61.3	73.9	**83.5**	**90.0**	**69.7**

Table 3. Recall@K and NMI across standard retrieval tasks. All methods are trained using ResNet50 for a fair comparison

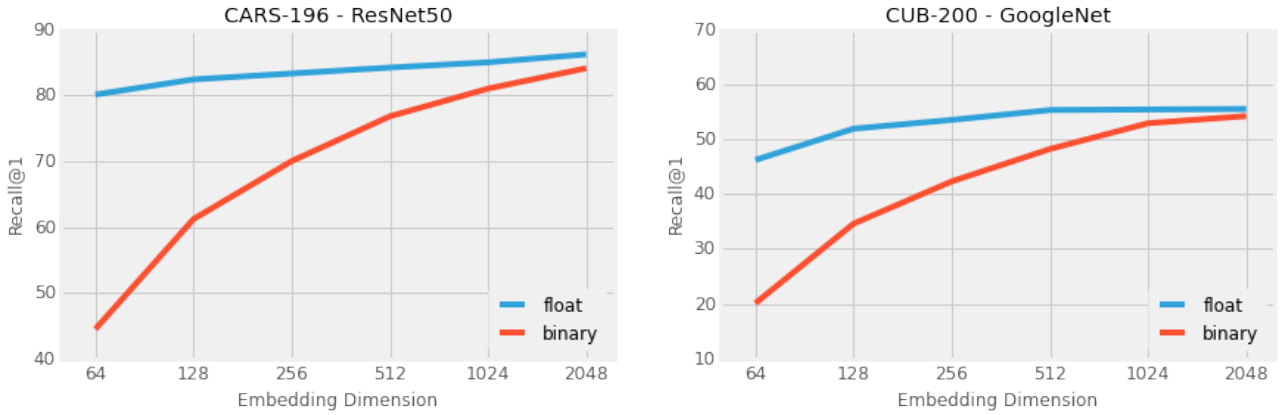

Figure 2. Recall@1 for CARS-196 (left) and CUB-200-2011 (right) across varying embedding dimensions. Softmax based embeddings improve performance when increasing dimensionality. The performance gap between float and binary embeddings converge when increasing dimensionality, showing that when given enough representational freedom, Softmax learns bit like features.

subsampling classes during training is an effective method of scaling softmax embedding training with little drop in performance. We also note that our class subsampling strategy is simply random selection. An interesting line of work in the future may be able to leverage the improvements in negative sampling [25] [28] for softmax class selection to further improve scalability and even performance.

6.6. Comparison against State of the Art

Finally we compare our best performing softmax embedding model against state-of-the-art metric learning approaches on Stanford Online Products, In-Shop, CARS-196, and CUB-200-2011. We train two variants of networks, BNInception with 1024 embedding dimensions and ResNet50 with 2048 dimensions. Because the dimensionality of the embedding is equal to the pooling feature of the

SPC	-	1	3	12	25	37	75
DCB	-	75	25	6	3	2	1
R@1	59.5	59.6	60.0	60.8	61.3	59.6	40.9

Table 4. ResNet50 Recall@1 on CUB-200-2011 dataset across varying samples per class for batch size of 75. (**SPC**) Samples per class in batch. (**DCB**) Distinct classes in batch. First column shows no class balancing in batch

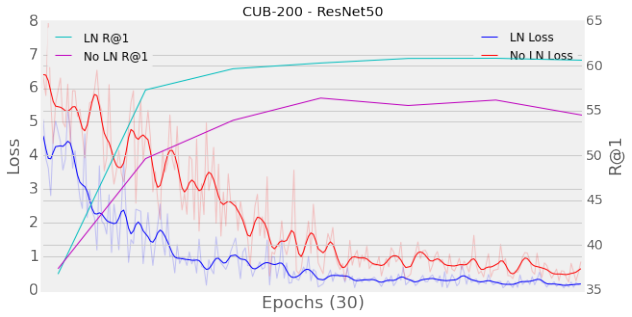

Figure 3. Loss and R@1 trends for training CUB-200 ResNet50 with and without Layer Normalization. Layer Normalization helps initialize learning, leading to better training convergence and R@1 performance.

Tmp	0.005	0.01	0.03	0.05	0.1	0.5	1.0
R@1	19.3	38.8	57.0	61.3	61.6	54.8	50.5

Table 5. ResNet50 Recall@1 on CUB-200-2011 dataset across varying temperature. (**Tmp**) Temperature.

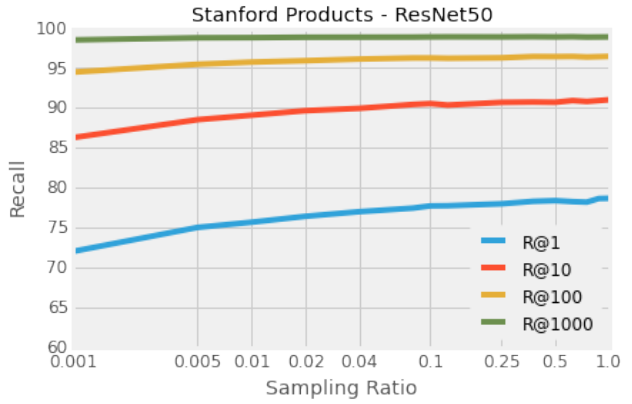

Figure 4. Recall@K for Stanford Online Products with ResNet50 across varying class sampling ratios used per iteration. We see that with sampling only 10% of classes per iteration (∼1K classes), we converge to a R@1 that is less than 1% absolute drop in performance from using all classes.

original network, we skip the fully connected layer and directly use the pooled feature for efficiency. We have verified this difference results in negligible performance changes (Stanford Online Products Recall@1 of 79.2 versus 79.5). Full results are presented in Table 6 and Table 7.

We see impressive retrieval performance from methods using only GoogleNet in Table 6 and Table 7. Looking at these approaches, we see that HDC [30] relies on three ensembles of GoogleNets with the final embedding size of 384 being the concatenation of three 128 dimensional embeddings. ABE-8 [27] modifies the internal GoogleNet architecture with attention modules to improve performance (not unexpected given precedence in classification architectures such as SeNet [10]). Finally A-BIER [18] where instead of a single fully connected layer on top of GoogleNet for the embedding, uses boosting through a cascade of smaller dimensional embeddings to learn decorrelated embeddings that when concatenated together, better leverages the higher dimensional space. We mark these methods as GoogleNet†.

As shown in Table 6 and Table 7, our 2048 dimensional ResNet50 embedding significantly outperforms previous approaches. Considering the higher dimensionality of our embeddings, we also show that our 2048 **binary** embedding, sharing the same memory footprint of a 64 dimensional float embedding, similarly significantly outperforms state-of-the-art baselines. These binary features were obtained by thresholding the float embedding features at zero as in Figure 2. Visualizations of the retrieval results for both the binary and float embeddings on Stanford Online Products is shown in Figure 1. Considering the scalability and performance of our binary softmax features along the simplicity of our approach, we believe softmax embeddings should be a strong baseline for future metric learning work.

7. Conclusion

In this paper, we have proposed a simple modification to the standard classification network which makes classification an effective metric learning approach. In the metric learning community, a diverse set of base networks for training embedding of different sizes are compared to one another. In our work, we conducted fair comparisons through extensive experimentation, and establish that softmax used in standard classification is a strong baseline in a wide variety of settings. Our source code will be released to help make bench-marking easier for future research.

We further propose that by adding temperature scaling and Layer Normalization, our approach can learn high-dimensional binary embeddings that beat state-of-the-art performance on image retrieval tasks with same or less memory footprint. The subsampling module we introduced makes our approach viable even for tasks with a very large number of lasses. Our approach not only address the concern of extreme classification, but most importantly establishes the validity of using classification as a state-of-the-art metric learning approach.

	Net.	SOP				In-Shop					
Recall@K		1	10	100	1000	1	10	20	30	40	50
Contrastive^{128} [22]	GoogleNet	42.0	58.2	73.8	89.1	-	-	-	-	-	-
Lifted Struct^{512} [22]	GoogleNet	62.1	79.8	91.3	97.4	-	-	-	-	-	-
Histogram Loss^{512} [24]	GoogleNet	63.9	81.7	92.2	97.7	-	-	-	-	-	-
Binomial Dev^{512} [24]	GoogleNet	65.5	82.3	92.3	97.6	-	-	-	-	-	-
Clustering^{64} [21]	BNInception	67.0	83.7	93.2	-	-	-	-	-	-	-
Npairs^{512} [20]	GoogleNet	67.7	83.8	93.0	97.8	-	-	-	-	-	-
HDC^{384} [30]	GoogleNet†	69.5	84.4	92.8	97.7	62.1	84.9	89.0	91.2	92.3	93.1
Angular Loss^{512} [13]	GoogleNet	70.9	85.0	93.5	98.0	-	-	-	-	-	-
Margin^{128} [28]	ResNet50	72.7	86.2	93.8	98.0	-	-	-	-	-	-
Proxy NCA^{64}[17]	BNInception	73.7	-	-	-	-	-	-	-	-	-
A-BIER^{512} [18]	GoogleNet†	74.2	86.9	94.0	97.8	83.1	95.1	96.9	97.5	97.8	98.0
HTL^{512} [25]	BNInception	74.8	88.3	94.8	98.4	-	-	-	-	-	-
HTL^{128} [25]	BNInception	-	-	-	-	80.9	94.3	95.8	97.2	97.4	97.8
ABE-8^{512} [27]	GoogleNet†	76.3	88.4	94.8	98.2	87.3	96.7	97.9	98.2	98.5	98.7
NormSoftmax^{1024}	BNInception	74.7	88.3	95.2	98.5	86.6	97.0	98.0	98.5	98.7	98.9
NormSoftmax^{128}	ResNet50	75.2	88.7	95.2	98.4	86.6	96.8	97.8	98.3	98.6	98.8
NormSoftmax^{512}	ResNet50	78.2	90.6	96.2	98.8	88.6	97.5	98.4	98.8	99.0	99.0
NormSoftmax^{2048}	ResNet50	**79.5**	**91.5**	**96.7**	**98.9**	**89.4**	**97.8**	**98.7**	**99.0**	**99.2**	**99.3**
NormSoftmax$^{2048\textbf{bits}}$	ResNet50	78.2	90.9	96.4	98.8	88.8	**97.8**	98.5	98.8	99.0	99.1

Table 6. Recall@K on Stanford Online Products (SOP) and In-Shop. GoogleNet† refers to non-conventional usage (Section 6.6)

	Net.	CARS-196				CUB-200			
Recall@K		1	2	4	8	1	2	4	8
Contrastive^{128} [22]	GoogleNet	21.7	32.3	46.1	58.9	26.4	37.7	49.8	62.3
Lifted Struct^{128} [22]	GoogleNet	49.0	60.3	72.1	81.5	47.2	58.9	70.2	80.2
Clustering^{64} [21]	BNInception	58.1	70.6	80.3	87.8	48.2	61.4	71.8	81.9
Histogram Loss^{512} [24]	GoogleNet	-	-	-	-	50.3	61.9	72.6	82.4
Binomial Dev^{512} [24]	GoogleNet	-	-	-	-	52.8	64.4	74.7	83.9
Npairs^{64} [20]	GoogleNet	71.1	79.7	86.5	91.6	51.0	63.3	74.3	83.2
Angular Loss^{512} [13]	GoogleNet	71.4	81.4	87.5	92.1	54.7	66.3	76.0	83.9
Proxy NCA^{64} [17]	BNInception	73.2	82.4	86.4	88.7	49.2	61.9	67.9	72.4
HDC^{384} [30]	GoogleNet†	73.7	83.2	89.5	93.8	53.6	65.7	77.0	85.6
Margin^{128} [28]	ResNet50	79.6	86.5	91.9	95.1	63.6	74.4	83.1	90.0
HTL^{512} [25]	BNInception	81.4	88.0	92.7	95.7	57.1	68.8	78.7	86.5
A-BIER^{512} [18]	GoogleNet†	82.0	89.0	93.2	96.1	57.5	68.7	78.3	86.2
ABE-8^{512} [27]	GoogleNet†	85.2	90.5	94.0	96.1	60.6	71.5	79.8	87.4
NormSoftMax^{1024}	BNInception	87.9	93.2	96.2	98.1	62.2	73.9	82.7	89.4
NormSoftmax^{128}	ResNet50	81.6	88.7	93.4	96.3	56.5	69.6	79.9	87.6
NormSoftmax^{512}	ResNet50	84.2	90.4	94.4	96.9	61.3	73.9	83.5	90.0
NormSoftmax^{2048}	ResNet50	**89.3**	**94.1**	**96.4**	**98.0**	**65.3**	**76.7**	**85.4**	**91.8**
NormSoftmax$^{2048\textbf{bits}}$	ResNet50	88.7	93.7	**96.4**	**98.0**	63.3	75.2	84.3	91.0

Table 7. Recall@K on CARS-196 and CUB-200-2011. GoogleNet† refers to non-conventional usage (Section 6.6)

References

[1] S. Bell and K. Bala. Learning visual similarity for product design with convolutional neural networks. *ACM Trans. on Graphics (SIGGRAPH)*, 34(4), 2015.

[2] K. S. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? In *Database Theory - ICDT '99, 7th International Conference, Jerusalem, Israel, January 10-12, 1999, Proceedings.*, pages 217–235, 1999.

[3] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification.

In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546. IEEE, 2005.

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[5] J. Goldberger, S. T. Roweis, G. E. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems 17 [Neural Information Processing Systems, NIPS 2004, December 13-18, 2004, Vancouver, British Columbia, Canada]*, pages 513–520, 2004.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[7] A. Hermans, L. Beyer, and B. Leibe. In Defense of the Triplet Loss for Person Re-Identification. Technical report.

[8] E. Hoffer and N. Ailon. Deep metric learning using triplet network. *CoRR*, abs/1412.6622, 2014.

[9] H. Hu, Y. Wang, L. Yang, P. Komlev, L. Huang, X. S. Chen, J. Huang, Y. Wu, M. Merchant, and A. Sacheti. Web-scale responsive visual search at bing. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 359–367, 2018.

[10] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. 2018.

[11] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[12] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[13] S. W. X. L. Y. L. Jian Wang, Feng Zhou. Deep metric learning with angular loss. In *International Conference on Computer Vision*, 2017.

[14] G. E. H. Jimmy Lei Ba, Jamie Ryan Kiros. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[15] Y. Jing, D. Liu, D. Kislyuk, A. Zhai, J. Xu, and J. Donahue. Visual search at pinterest. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*.

[16] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

[17] Y. Movshovitz-Attias, A. Toshev, T. K. Leung, S. Ioffe, and S. Singh. No fuss distance metric learning using proxies. *CoRR*, abs/1703.07464, 2017.

[18] M. Opitz, G. Waltner, H. Possegger, and H. Bischof. Deep Metric Learning with BIER: Boosting Independent Embeddings Robustly. *arXiv:cs/1801.04815*, 2018.

[19] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[20] K. Sohn. Improved deep metric learning with multi-class n-pair loss objective. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1857–1865. Curran Associates, Inc., 2016.

[21] H. O. Song, S. Jegelka, V. Rathod, and K. Murphy. Deep metric learning via facility location. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[22] H. O. Song, Y. Xiang, S. Jegelka, and S. Savarese. Deep metric learning via lifted structured feature embedding. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[23] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014.

[24] E. Ustinova and V. Lempitsky. Learning deep embeddings with histogram loss. In *Neural Information Processing Systems*, 2016.

[25] D. D. Weifeng Ge, Weilin Huang and M. R. Scott. Deep metric learning with hierarchical triplet loss. In *ECCV*, 2018.

[26] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

[27] K. C. J. L. K. K. Wonsik Kim, Bhavya Goyal. Attention-based ensemble for deep metric learning. In *ECCV*, 2018.

[28] C. Wu, R. Manmatha, A. J. Smola, and P. Krähenbühl. Sampling matters in deep embedding learning. *CoRR*, abs/1706.07567, 2017.

[29] Z. Wu, A. A. Efros, and S. X. Yu. Improving generalization via scalable neighborhood component analysis. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VII*, pages 712–728, 2018.

[30] Y. Yuan, K. Yang, and C. Zhang. Hard-aware deeply cascaded embedding. *arXiv preprint arXiv:1611.05720*, 2016.

[31] J. W. D. T. C. R. Yushi Jing, Henry Rowley and M. Covell. Google image swirl: a large-scale content-based image visualization system. In *Proceedings of the 21st International Conference on World Wide Web*, 2012.

[32] A. Zhai, D. Kislyuk, Y. Jing, M. Feng, E. Tzeng, J. Donahue, Y. L. Du, and T. Darrell. Visual discovery at pinterest. *arXiv preprint arXiv:1702.04680*, 2017.

[33] Y. Zhang, P. Pan, Y. Zheng, K. Zhao, Y. Zhang, X. Ren, and R. Jin. Visual search at alibaba. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 993–1001, 2018.

[34] S. Q. X. W. Ziwei Liu, Ping Luo and X. Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.