Sampling Matters in Deep Embedding Learning

Chao-Yuan Wu*
UT Austin
cywu@cs.utexas.edu

R. Manmatha
A9/Amazon
manmatha@a9.com

Alexander J. Smola
Amazon
smola@amazon.com

Philipp Krähenbühl
UT Austin
philkr@cs.utexas.edu

Abstract

Deep embeddings answer one simple question: How similar are two images? Learning these embeddings is the bedrock of verification, zero-shot learning, and visual search. The most prominent approaches optimize a deep convolutional network with a suitable loss function, such as contrastive loss or triplet loss. While a rich line of work focuses solely on the loss functions, we show in this paper that selecting training examples plays an equally important role. We propose distance weighted sampling, which selects more informative and stable examples than traditional approaches. In addition, we show that a simple margin based loss is sufficient to outperform all other loss functions. We evaluate our approach on the Stanford Online Products, CAR196, and the CUB200-2011 datasets for image retrieval and clustering, and on the LFW dataset for face verification. Our method achieves state-of-the-art performance on all of them.

1. Introduction

Models that transform images into rich, semantic representations lie at the heart of modern computer vision, with applications ranging from zero-shot learning [5, 41] and visual search [3, 10, 22, 29], to face recognition [6, 23, 25, 28] or fine-grained retrieval [22, 28, 29]. Deep networks trained to respect pairwise relationships have emerged as the most successful embedding models [4, 6, 25, 34].

The core idea of deep embedding learning is simple: pull similar images closer in embedding space and push dissimilar images apart. For example, the contrastive loss [11] forces all positives images to be close, while all negatives should be separated by a certain fixed distance. However, using the same fixed distance for all images can be quite restrictive, discouraging any distortions in the embedding space. This motivated the triplet loss, which only requires negative images to be farther away than any positive images on a per-example basis [25]. This triplet loss is currently among the best-performing losses on standard embedding tasks [22, 25, 45]. Unlike pairwise losses, the triplet loss does not just change the loss function in isolation, it changes the way positive and negative example are selected. This provides us with two knobs to turn: the loss and the sampling strategy. See Figure 1 for an illustration.

In this paper, we show that sample selection in embedding learning plays an equal or more important role than the loss. For example, different sampling strategies lead to drastically different solutions for the same loss function. At the same time many different loss functions perform similarly under a good sampling strategy: A contrastive loss works almost as well as the triplet loss, if the two use the same sampling strategy. In this paper, we analyze existing sampling strategies, and show why they work and why not. We then propose a new sampling strategy, where samples are drawn uniformly according to their relative distance from one another. This corrects the bias induced by the geometry of embedding space, while at the same time ensuring any data point has a chance of being sampled. Our proposed sampling leads to a lower variance of gradients, and thus stabilizes training, resulting in a qualitatively better embedding irrespective of the loss function.

Loss functions obviously also matter. We propose a simple margin-based loss as an extension to the contrastive loss. It only encourages all positive samples to be within a distance of each other rather than being as close as possible. It relaxes the loss, making it more robust. In addition, by using isotonic regression, our margin based loss focuses on the relative orders instead of absolute distances.

Our margin based loss and distance weighted sampling achieve state-of-the-art image retrieval and clustering performance on the Stanford Online Products, CARS196, and the CUB200-2011 datasets. It also outperforms previous state-of-the-art results on the LFW face verification dataset [16] using standard publicly available training data. Both our loss function and sampling strategy are easy to implement and efficient to train.

*Part of this work performed while interning at Amazon.

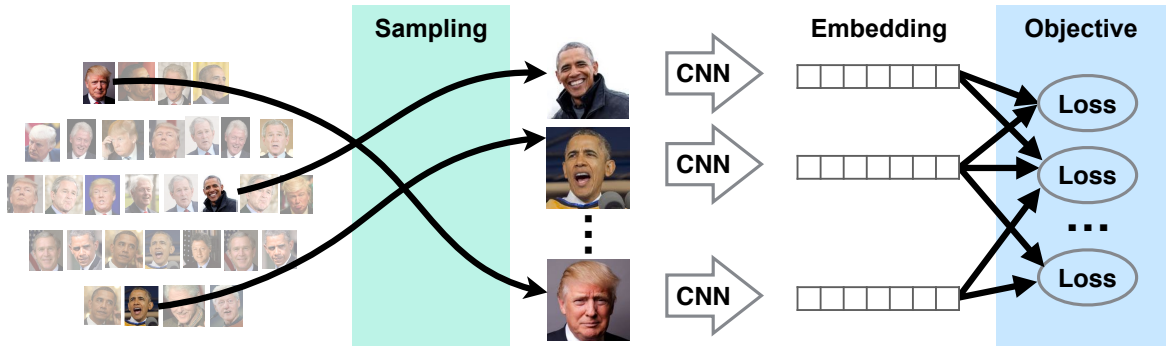


Figure 1: An overview of deep embedding learning: The first stage samples images and forms a batch. A deep network then transforms the images into embeddings. Finally, a loss function measures the quality of our embedding. Note that both the sampling and the loss function influence the overall training objective.

2. Related Work

The idea of using neural networks to extract features that respect certain relationships dates back to the 90s. *Siamese Networks* [4] find an embedding space such that similar examples have similar embeddings and vice versa. Such networks are trained end-to-end, sharing weights between all mappings. Siamese Networks were first applied to signature verification, and later extended to face verification and dimensionality reduction [6, 11]. However, given the limited compute power at the time and their non-convex nature, these approaches initially did not enjoy much attention. Convex approaches were much more popular [7, 39]. For example, the triplet loss [26, 36] is one of the most prominent methods that emerged from convex optimization.

Given sufficient data and computational power both schools of thought were combined into a *Siamese* architecture using triplet losses. This leads to near human performance in face verification [23, 25]. Motivated by the triplet loss, some enforce constraints on even more examples. For example, PDDM [15] and Histogram Loss [34] use quadruplets. Beyond that, the n-pair loss [28] and Lifted Structure [22] defines constraints on all images in a batch.

This plethora of loss functions is quite reminiscent of the ranking problem in information retrieval. There a combination of individual, pair-wise [14], and list-wise approaches [35] are used to maximize relevance. Of note is isotonic regression which disentangles the pairwise comparisons for greater computational efficiency. See [21] for an overview.

Some papers explore modeling of other properties. Structural Clustering [29] optimizes for clustering quality. PDDM [15] proposes a new module to model local feature structure. HDC [41] trains an ensemble to model examples of different "hard levels". In contrast, here we show that a simple pairwise loss is sufficient if paired with the right sampling strategy.

Example selection techniques are relatively less studied. For the contrastive loss it is common to select from all possible pairs at random [3, 6, 11], and sometimes with hard negative mining [27]. For the triplet loss, semi-hard negative mining, first used in FaceNet [25], is widely adopted [22,23]. Sampling has been studied for stochastic optimization [43] with the goal of accelerating convergence to the same global loss function. In contrast, in embedding learning the sampling actually changes the overall loss function considered. In this paper we show how sampling affects the real-world performance of deep embedding learning.

3. Preliminaries

Let $f(x_i)$ be an embedding of a datapoint $x_i \in \mathbb{R}^N$, where $f : \mathbb{R}^N \to \mathbb{R}^D$ is a differentiable deep network with parameters Θ. Often $f(x_i)$ is normalized to have unit length for training stability [25]. Our goal is to learn an embedding that keeps similar data points close, while pushing dissimilar datapoints apart. Formally we define the distance between two datapoints as $D_{ij} := \|f(x_i) - f(x_j)\|$, where $\|\cdot\|$ denotes the Euclidean norm. For any positive pair of datapoints $y_{ij} = 1$ this distance should be small, and for negative pair $y_{ij} = 0$ it should be large.

The contrastive loss directly optimizes this distance by encouraging all positive distances to approach 0, while keeping negative distances above a certain threshold:

$$\ell^{\text{contrast}}(i,j) := y_{ij} D_{ij}^2 + (1 - y_{ij}) \left[\alpha - D_{ij}\right]_+^2 .$$

One drawback of the contrastive loss is that we have to select a constant margin α for all pairs of negative samples. This implies that visually diverse classes are embedded in the same small space as visually similar ones. The embedding space does not allow for distortions.

In contrast the triplet loss merely tries to keep all positives closer to any negatives for each example:

$$\ell^{\text{triplet}}(a,p,n) := \left[D_{ap}^2 - D_{an}^2 + \alpha\right]_+ .$$

This formulation allows the embedding space to be arbitrarily distorted and does not impose a constant margin α.

From the risk minimization perspective, one might aim at optimizing the aggregate loss over all $O(n^2)$ pairs or $O(n^3)$ triples respectively. That is

$$R^{(\cdot)} := \sum_{t \in \{\text{all pairs/triplets}\}} \ell^{(\cdot)}(t).$$

This is computationally infeasible. Moreover, once the network converges, most samples contribute in a minor way as very few of the negative margins are violated.

This lead to the emergence of many heuristics to accelerate convergence. For the contrastive loss, hard negative mining usually offers faster convergence. For the triplet loss, it is less obvious, as hard negative mining often leads to collapsed models, i.e. all images have the same embedding. FaceNet [25] thus proposed to use a somewhat mysterious *semi-hard* negative mining: given an anchor a and a positive example p, obtain a negative instance n via

$$n^\star_{ap} := \operatorname*{argmin}_{n: D(a,n) > D(a,p)} D_{an},$$

within a batch. This yields a violating example that is fairly hard but not too hard. Batch construction also matters. In order to obtain more informative triplets, FaceNet uses a batch size of 1800 and ensures that each identity has roughly 40 images in a batch [25]. Even how to best select triplets within a batch is unclear. Parkhi *et al.* [23] use online selection, so that only one triplet is sampled for every (a, p) pair. OpenFace [2] employs offline triplet selection, so that a batch has $1/3$ of images as anchors, positives, and negatives respectively.

In short, sampling matters. It implicitly defines a rather heuristic objective function by weighting samples. Such an approach makes it hard to reproduce and extend the insights to different datasets, different optimization frameworks or different architectures. In the next section, we analyze some of these techniques, and explain why they offer better results. We then propose a new sampling strategy that outperforms current state of the art.

4. Distance Weighted Margin-Based Loss

To understand what happens when sampling negative uniformly, recall that our embeddings are typically constrained to the n-dimensional unit sphere $\mathbb{S}^{n-1}$ for large $n \geq 128$. Consider the situation where the points are uniformly distributed on the sphere. In this case, the distribution of pairwise distances follows

$$q(d) \propto d^{n-2} \left[1 - \tfrac{1}{4} d^2\right]^{\frac{n-3}{2}}.$$

See [1] for a derivation. Figure 2 shows concentration of measure occurring. In fact, in high dimensional space, $q(d)$ approaches $\mathcal{N}(\sqrt{2}, \frac{1}{2n})$. In other words, if negative examples are scattered uniformly, and we sample them randomly,

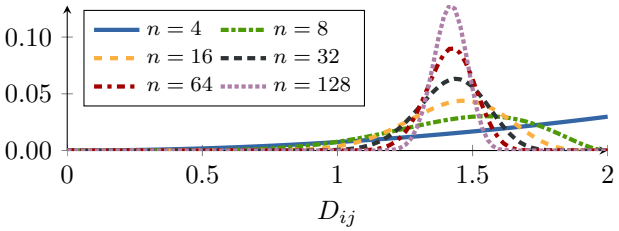


Figure 2: Density of datapoints on the D-dimensional unit sphere. Note the concentration of measure as the dimensionality increases — most points are almost equidistant.

we are likely to obtain examples that are $\sqrt{2}$-away. For thresholds less than $\sqrt{2}$, this induces no loss, and thus no progress for learning. Learned embeddings follow a very similar distribution, and thus the same reasoning applies. See supplementary material for details.

Sampling negative examples that are too hard causes a different issue. Consider a negative pair $t := (a, n)$ or a triplet $t := (a, p, n)$. The gradient with respect to the negative example $f(x_n)$ is in the form of

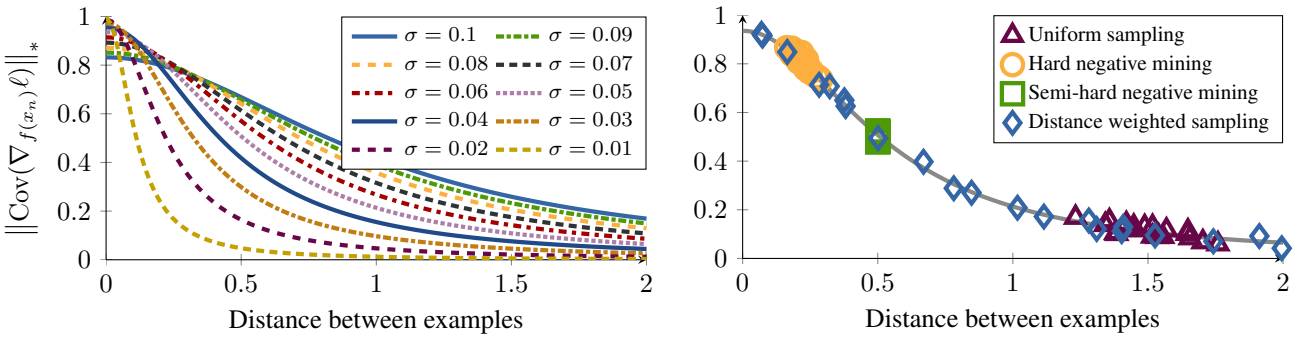
$$\partial_{f(x_n)} \ell^{(\cdot)} = \frac{h_{an}}{\|h_{an}\|} w(t)$$

for some function $w(\cdot)$ and $h_{an} := f(x_a) - f(x_n)$. Note that the first term $\frac{h_{an}}{\|h_{an}\|}$ determines the direction of the gradient. A problem arises when $\|h_{an}\|$ is small, and our estimates of embedding are noisy. Given enough noise z introduced by the training algorithm, direction $\frac{h_{an}+z}{\|h_{an}+z\|}$ is dominated by noise. Figure 3a shows the nuclear norm of the covariance matrix for the direction of gradient with $z \sim \mathcal{N}\left(0, \sigma^2 I\right)$. We can see that when negative examples are too close/hard, the gradient has high variance and it has low signal to noise ratio. At the same time random samples are often too far apart to yield a good signal.

Distance weighted sampling. We thus propose a new sampling distribution that corrects the bias while controlling the variance. Specifically, we sample uniformly according to distance, i.e. sampling with weights $q(d)^{-1}$. This gives us examples which are spread out instead of being clustered around a small region. To avoid noisy samples, we clip the weighted sampling. Formally, given an anchor example a, distance weighted sampling samples negative pair $(a, n^\star)$ with

$$\Pr\left(n^\star = n | a\right) \propto \min\left(\lambda, q^{-1}\left(D_{an}\right)\right).$$

Figure 3b compares the simulated examples drawn from different strategies along with their variance of gradients. Hard negative mining always offers examples in the high-variance region. This leads to noisy gradients that cannot effectively push two examples apart, and consequently a



(a) Variance of gradient at different noise levels.

(b) Sample distribution for different strategies.

Figure 3: (a) shows the nuclear norm of a noisy gradient estimate for various levels of noise. High variance means the gradient is close to random, while low variance implies a deterministic gradient estimate. Lower is better. Note that higher noise levels have a lower variance at distance 0. This is due to the spherical projection imposed by the normalization. (b) shows the empirical distribution of samples drawn for different strategies. Distance weighted sampling selects a wide range of samples, while all other approaches are biased towards certain distances.

collapsed model. Random sampling yields only easy examples that induce no loss. Semi-hard negative mining finds a narrow set in between. While it might converge quickly at the beginning, at some point no examples are left within the band, and the network will stop making progress. FaceNet reports a consistent finding: the decrease of loss slows down drastically after some point, and their final system took 80 days to train [25]. Distance weighted sampling offers a wide range of examples, and thus steadily produce informative examples while controlling the variance. In Section 5, we will see that distance weighted sampling brings performance improvements in almost all loss functions tested. Of course sampling only solves half of the problem, but it puts us in a position to analyze various loss functions.

Figure 4a and Figure 4b depict the contrastive loss and the triplet loss. There are two key differences, which in general explain why the triplet loss outperforms contrastive loss: The triplet loss does not assume a predefined threshold to separate similar and dissimilar images. Instead, it enjoys the flexibility to distort the space to tolerate outliers, and to adapt to different levels of intra-class variance for different classes. Second, the triplet loss only requires positive examples to be closer than negative examples, while the contrastive loss spends efforts on gathering all positive examples as close together as possible. The latter is not necessary. After all, maintaining correct relative relationship is sufficient for most applications, including image retrieval, clustering, and verification.

On the other hand, in Figure 4b we also observe the concave shape of the loss function for negative examples in the triplet loss. In particular, note that for hard negatives (with small D_{an}), the gradient with respective to negative example is approaching zero. It is not hard to see why hard negative mining results in a collapsed model in this case: it gives large attracting gradients from hard positive pairs, but small repelling gradients from hard negative pairs, so all points are eventually gathered to the same point. To make the loss stable for examples from all distances, one simple remedy is to use ℓ_2 instead of ℓ_2^2, i.e.

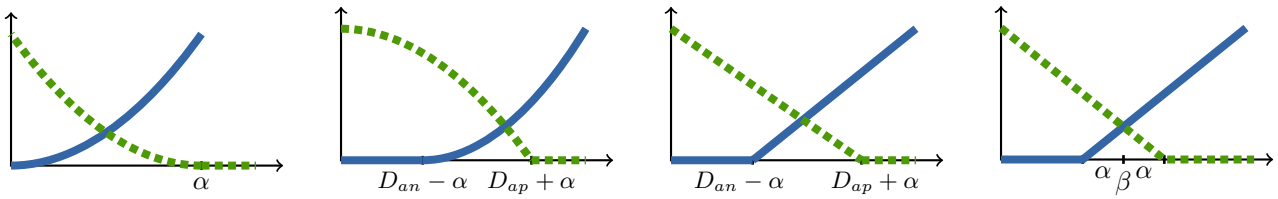
$$\ell^{\text{triplet},\ell_2} := (D_{ap} - D_{an} + \alpha)_+ .$$

Figure 4c presents the loss function. Now its gradients with respect to any embedding $f(x)$ will always have length one. See e.g. [12, 20] for more discussions about the benefits of using gradients of a fixed length. This simple fix together with distance weighted sampling already outperforms the traditional ℓ_2^2 triplet loss, as shown in Section 5.

Margin based loss. These observations motivate our design of a loss function which enjoys the flexibility of the triplet loss, has a shape suitable for examples from all distances, while offering the computational efficiency of a contrastive loss. The basic idea can be traced back to the insight that in ordinal regression only the relative order of scores matters [17]. That is, we only need to know the crossover between both sets. Isotonic regression exploits this by estimating such a threshold separately and then penalizes scores relative to the threshold. We use the same trick, now applied to pairwise distances rather than score functions. The adaptive margin based loss is defined as

$$\ell^{\text{margin}}(i,j) := (\alpha + y_{ij}(D_{ij} - \beta))_+ .$$

Here β is a variable that determines the boundary between positive and negative pairs, α controls the margin of separation, and $y_{ij} \in \{-1, 1\}$. Figure 4d visualizes this new loss function. We can see that it relaxes the constraint on positive examples from contrastive loss. It effectively imposes a



(a) Contrastive loss [11] (b) Triplet loss ℓ_2^2 [25] (c) Triplet loss ℓ_2 (d) Margin based loss

Figure 4: Loss vs. pairwise distance. The solid blue lines show the loss function for positive pairs, the dotted green for negative pairs. Our loss finds an optimal boundary β between positive and negative pairs, and α ensures that they are separated by a large margin.

large margin loss on the shifted distance $D_{ij} - \beta$. This loss is very similar to a support vector classifier (SVC) [8].

To enjoy the flexibility as a triplet loss, we need a more flexible boundary parameter β which depends on class-specific $\beta^{(\text{class})}$ and example-specific $\beta^{(\text{img})}$ terms.

$$\beta(i) := \beta^{(0)} + \beta_{c(i)}^{(\text{class})} + \beta_i^{(\text{img})}$$

In particular, the example-specific offset $\beta_i^{(\text{img})}$ plays the same role as the threshold in a triple loss. It is infeasible to manually select all the $\beta_c^{(\text{class})}$s and $\beta_i^{(\text{img})}$s. Instead, we would like to jointly learn these parameters. Fortunately, the gradient of β can be easily calculated as

$$\partial_\beta \ell^{\text{margin}}(i,j) = -y_{ij}\mathbf{1}\left\{\alpha > y_{ij}(\beta - D_{ij})\right\}$$

It is clear that larger values of β are more desirable, since they amount to a better use of the embedding space. Hence, to regularize β, we incorporate a hyperparameter ν, and it leads to the optimization problem

$$\text{minimize} \sum_{(i,j)} \ell^{\text{margin}}(i,j) + \nu\left(\beta^{(0)} + \beta_{c(i)}^{(\text{class})} + \beta_i^{(\text{img})}\right)$$

Here ν adjusts the difference between the number of points that violate the margin on the left and on the right. This can be seen by observing that their gradients need to cancel out at an optimal β. Note that the use of ν here is very similar to the ν-trick in ν-SVM [24].

Relationship to isotonic regression. Optimizing the margin based loss can be viewed as solving a ranking problem for distances. Technically it shares similarity with learning-to-rank problems in information retrieval [21, 44]. To see this first note at optimal β, the empirical risk can be written as

$$R^{\text{margin}} := \min_\beta \sum_{(i,j)} \left(\alpha + y_{ij}(D_{ij} - \beta)\right)_+.$$

One can show that $R^{\text{margin}} = \sum_{(i,j)} \xi_{ij}^*$, where ξ^*s are the solution to

$$\text{minimize} \sum_{(i,j)\in\mathcal{X}^{\text{pos}}} \xi_{ij} + \sum_{(k,l)\in\mathcal{X}^{\text{neg}}} \xi_{kl}$$

subject to

$$D_{kl} + \xi_{kl} - D_{ij} + \xi_{ij} \geq 2\alpha, (i,j) \in \mathcal{X}^{\text{pos}}, (k,l) \in \mathcal{X}^{\text{neg}}$$
$$\xi_{ij}, \xi_{kl} \geq 0,$$

where $\mathcal{X}^{\text{pos}} := \{(i,j) : y_{ij} = 1\}$, and $\mathcal{X}^{\text{neg}} := \{(i,j) : y_{ij} = -1\}$. This is an isotonic regression defined on absolute error. We see that the margin based loss is the amount of "minimum-effort" updates to maintain relative orders. It focuses on the relative relationships, i.e. focusing on the separation of positive-pair distances and the negative-pair distances. This is in contrast to traditional loss functions such as the contrastive loss, where losses are defined relative to a predefined threshold.

5. Experiments

We evaluate our method on image retrieval, clustering and verification. For image retrieval and clustering, we use the Stanford Online Products [22], CARS196 [19], and the CUB200-2011 [37] datasets, following the experimental setup of Song *et al.* [22]. The Stanford Online Product dataset contains 120,053 images of 22,634 categories. The first 11,318 categories are used for training, and the remaining are used for testing. The CARS196 dataset contains 16,185 car images of 196 models. We use the first 98 models for training, and the remaining for testing. The CUB200-2011 dataset contains 11,788 bird images of 200 species. The first 100 species are used for training, the remainder for testing.

We evaluate the quality of image retrieval based on the standard Recall@k metric, following Song *et al.* [22]. We use NMI score, $I(\Omega,\mathbb{C})/\sqrt{H(\Omega)H(\mathbb{C})}$, to evaluate the quality of clustering alignments $\mathbb{C} = \{c_1, \ldots, c_n\}$, given a ground-truth clustering $\Omega = \{\omega_1, \ldots, \omega_n\}$. Here $I(\cdot,\cdot)$ and $H(\cdot)$ denotes mutual information and entropy respectively. We use K-means algorithm for clustering.

For verification, we train our model on the largest publicly available face dataset, CASIA-WebFace [40], and evaluate on the standard LFW [16] dataset. The VGG face dataset [23] is bigger, but many of its links have expired. The CASIA-WebFace dataset contains 494,414 images of 10,575 people. The LFW dataset consists of 13,233 images of 5,749 people. Its verification benchmark contains 6,000 verification pairs, split into 10 subsets. We select the verification threshold for one split based on the remaining nine splits.

Unless stated otherwise, we use an embedding size of 128 and an input image size of 224×224 in all experiments. All models are trained using Adam [18] with a batch size of 200 for face verification, 80 for Stanford Online Products, and 128 for other experiments. The network architecture follows ResNet-50 (pre-activation) [13]. To accelerate training, we use a simplified version of ResNet-50 in the face verification experiments. Specifically, we use only $64, 96, 192, 384, 768$ filters in the 5 stages respectively, instead of the originally proposed $64, 256, 512, 1024, 2048$ filters. We did not observe any obvious performance degradations due to the change. Horizontal mirroring and random crops from 256×256 are used for data augmentation. During testing we use a single center crop. Face images are aligned by MTCNN [42]. When alignment fails, we use a center crop. Following FaceNet [25], we use $\alpha = 0.2$, and for the margin based loss we initialize $\beta^{(0)} = 1.2$ and $\beta^{(\text{class})} = \beta^{(\text{img})} = 0$.

Note that some previous papers use the provided bounding boxes while others do not. To fairly compare with previous methods, we evaluate our methods on both the original images and the ones cropped by bounding boxes. For the CARS196 dataset we scale the cropped images to 256×256. For CUB200, we scale and pad the images such that their longer side is 256 pixels, keeping the aspect ratio fixed.

Our batch construction follows FaceNet [25]. We use $m = 5$ positive images per class in a batch. All positive pairs within a batch are sampled. For each example in a positive pair, we sample one negative pair. This ensures that the number of positive and negative pairs are balanced, and every example belongs to the same number of positive pairs and the same number of negative pairs.

5.1. Ablation study

We start by understanding the effect of the loss function, the adaptive margin and the specific functional choice. We focus on Stanford Online Products, as it is the largest among the three image retrieval datasets. Note that image retrieval favors triplet losses over contrastive losses, since only relative relationships matter. Here all models are trained from scratch. Since different methods converge at different rates, we train all methods for 100 epochs, and report the performance at their best epoch rather than at the end of training.

k	1	10	100	1000
Random				
Contrastive loss [11]	30.1	51.6	72.3	88.4
Margin	37.5	56.3	73.8	88.3
Semi-hard				
Contrastive loss [11]	49.4	67.4	81.8	92.1
Triplet ℓ_2^2 [25]	49.7	68.1	82.5	92.9
Triplet ℓ_2	47.4	67.5	83.1	93.6
Margin	<u>61.0</u>	<u>74.6</u>	85.3	93.6
Distance weighted				
Contrastive loss [11]	39.2	60.8	79.1	92.2
Triplet ℓ_2^2 [25]	53.4	70.8	83.8	93.4
Triplet ℓ_2	54.5	72.0	<u>85.4</u>	**94.4**
Margin	**61.7**	**75.5**	**86.0**	<u>94.0</u>
Margin (pre-trained)	72.7	86.2	93.8	98.0

Table 1: Recall@k evaluated on Stanford Online Products. The bold numbers indicate the best and the underlined numbers indicate the second best performance.

We compare random sampling and semi-hard negative mining to our distance weighted sampling. For semi-hard sampling, there is no natural choice of a distance lower bound for pairwise loss functions. In this experiment we use a lower bound of 0.5 to simulate the positive distance in triplet loss. We consider the contrastive loss, the triplet loss and our margin based loss. By random sampling, we refer to uniform sampling from all positive and negative pairs. Since such a definition is not applicable for triplet losses, we test only the contrastive and margin based losses.

Results are presented in Table 1. We see that given the same loss function, different sampling distributions lead to very different performance. In particular, while the contrastive loss yields considerably worse results than triplet loss with random sampling, its performance significantly improves when using a sampling procedure similar to triplet loss. This evidence disproves a common misunderstanding of contrastive loss vs. triplet loss: the strength of triplet loss comes not just from the loss function itself, but more importantly from the accompanying sampling methods. In addition, distance weighted sampling consistently offers a performance boost for almost all loss functions. The only exception is the contrastive loss. We found it to be very sensitive to its hyperparameters. While we found good hyperparameters for random and semi-hard sampling, we were not able to find a well-performing hyperparameter for the distance weighted sampling yet. On the other hand, margin based loss automatically learns a suitable offset β and trains well. Notably, the margin based loss outperforms other loss functions by a large margin irrespective of sampling strategies. These observations hold with multiple batch sizes, as shown in Table 2. We also try pre-training our model using ILSVRC 2012-CLS [9] dataset, as is commonly done in

Loss, batch size	Random	Semi-hard	Dist. weighted
Triplet ℓ_2, 40	-	44.3	52.9
Triplet ℓ_2, 80	-	47.4	54.5
Triplet ℓ_2, 120	-	48.8	54.7
Margin, 40	41.9	60.7	61.1
Margin, 80	37.5	61.0	**61.7**
Margin, 120	37.7	59.6	60.5

Table 2: Recall@1 evaluated on Stanford Online Products for various batch sizes (40, 80, 120). Distance weighted sampling consistently outperforms other sampling strategies irrespective of the batch size. See supplementary material for Recall@10, 100, and 1000.

Figure 5: Retrieval results for randomly chosen query images in Stanford Online Products. Our loss retrieves more relevant images.

prior work [3, 22]. Pre-training offers a 10% boost in recall. In the following sections we focus on pre-trained models for fair comparison.

Next, we qualitatively evaluate these methods. Figure 5 presents the retrieval results on randomly picked query images. We can see that triplet loss generally offers reasonable results, but makes mistakes in some cases. On the other hand, our method gives much more accurate results.

To evaluate the gains obtained by learning a flexible boundary β, we compare models using a fixed β to models using learned βs. The results are summarized in Table 3. We see that the use of more flexibly class-specific $\beta^{(\text{class})}$ indeed offers advantages over various values of fixed $\beta^{(0)}$. We also test using example-specific $\beta^{(\text{img})}$, but the experiments are inconclusive. We conjecture that learning example-specific $\beta^{(\text{img})}$ might have introduced too many parameters and caused over-fitting.

Convergence speed. We further analyze the effects of sampling on the convergence speed. We compare margin based loss using distance weighted sampling with the two most commonly used deep embedding approaches: triplet loss with semi-hard sampling and contrastive loss with random sampling. The learning curves are shown in Figure 6. We see that triplet loss trained with semi-hard negative mining converges slower as it ignores too many examples. Contrastive loss with random sampling converges even slower. Distance weighted sampling, which uses more informative and stable examples, converges faster and more accurately.

k	1	10	100	1000
Fixed $\beta^{(0)} = 0.8$	61.3	79.2	90.5	97.0
Fixed $\beta^{(0)} = 1.0$	70.4	84.6	93.1	97.8
Fixed $\beta^{(0)} = 1.2$	71.1	85.1	93.2	97.8
Fixed $\beta^{(0)} = 1.4$	67.1	82.6	92.2	97.7
Learned $\beta^{(\text{class})}$	**72.7**	**86.2**	**93.8**	**98.0**

Table 3: Recall@k on Stanford Online Products for margin based loss with fixed and learned β. Results at 8K iterations are reported. The values of learned $\beta_c^{(\text{class})}$ range from 0.94 to 1.45, hence our choice for a consensus value of $\beta^{(0)}$.

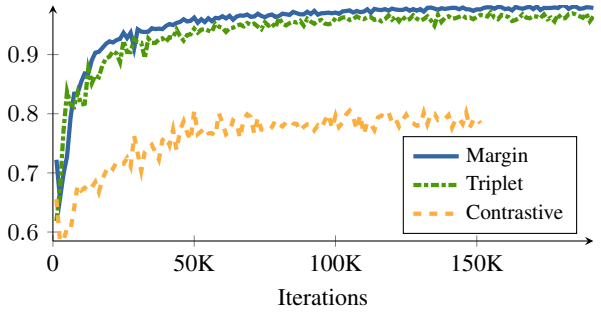


Figure 6: Validation accuracy curve, trained on CASIA-WebFace and evaluated on LFW. The margin based loss with distance weighted sampling converges quickly and stably, outperforming other methods.

Time complexity of sampling The computational cost of sampling is negligible. On a Tesla P100 GPU, forward and backward pass take about 0.55 second per batch (size 120). Sampling takes only 0.00031 second with semi-hard sampling and 0.0043 second with distance weighted sampling, even with our single-thread CPU implementation. Both strategies take $\mathcal{O}\left(nm(n-m)\right)$, where n is the batch size, and m is the number of images per class in a batch.

5.2. Quantitative Results

We now compare our approach to other state-of-the-art methods. Image retrieval and clustering results are summarized in Table 4, 5 and 6. We can see that our model achieves the best performance in all three datasets. In particular, margin based loss outperforms extensions of triplet loss, such as LiftedStruct [22], StructClustering [29], N-pairs [28], and PDDM [15]. It also outperforms histogram loss [34], which requires computing similarity histograms. Also note that

k	1	10	100	1000	NMI
Histogram [34]	63.9	81.7	92.2	97.7	-
Binomial Deviance [34]	65.5	82.3	92.3	97.6	-
Triplet Semi-hard [25, 29]	66.7	82.4	91.9	-	89.5
LiftedStruct [22, 29]	62.5	80.8	91.9	-	88.7
StructClustering [29]	67.0	83.7	93.2	-	89.5
N-pairs [28]	67.7	83.8	93.0	97.8	88.1
HDC [41]	69.5	84.4	92.8	97.7	-
Margin	**72.7**	**86.2**	**93.8**	**98.0**	**90.7**

Table 4: Recall@k and NMI on Stanford Online Products [22].

k	1	2	4	8	16	NMI
			Original Images			
Triplet Semi-hard [25, 29]	51.5	63.8	73.5	82.4	-	53.4
LiftedStruct [22, 29]	53.0	65.7	76.0	84.3	-	56.9
StructClustering [29]	58.1	70.6	80.3	87.8	-	59.0
N-pairs [28]	71.1	79.7	86.5	91.6	-	64.0
HDC [41]	73.7	83.2	89.5	93.8	96.7	-
Margin	**79.6**	**86.5**	**91.9**	**95.1**	**97.3**	**69.1**
			Cropped Images			
PDDM Triple [15]	46.4	58.2	70.3	80.1	88.6	-
PDDM Quadruplet [15]	57.4	68.6	80.1	89.4	92.3	-
HDC [41]	83.8	89.8	93.6	96.2	97.8	-
Margin	**86.9**	**92.7**	**95.6**	**97.6**	**98.7**	**77.5**

Table 5: Recall@k and NMI on CARS196 [19].

k	1	2	4	8	16	NMI
			Original Images			
Histogram [34]	52.8	64.4	74.7	83.9	90.4	-
Binomial Deviance [34]	50.3	61.9	72.6	82.4	88.8	-
Triplet [25, 29]	42.6	55.0	66.4	77.2	-	55.4
LiftedStruct [22, 29]	43.6	56.6	68.6	79.6	-	56.5
Clustering [29]	48.2	61.4	71.8	81.9	-	59.2
N-pairs [28]	51.0	63.3	74.3	83.2	-	60.4
HDC [41]	53.6	65.7	77.0	85.6	91.5	-
Margin	**63.6**	**74.4**	**83.1**	**90.0**	**94.2**	**69.0**
			Cropped Images			
PDDM Triplet [15]	50.9	62.1	73.2	82.5	91.1	-
PDDM Quadruplet [15]	58.3	69.2	79.0	88.4	93.1	-
HDC [41]	60.7	72.4	81.9	89.2	93.7	-
Margin	**63.9**	**75.3**	**84.4**	**90.6**	**94.8**	**69.8**

Table 6: Recall@k and NMI on CUB200-2011 [37].

our model uses only one 128-dimensional embedding for each image. This is much more concise and simpler than HDC [41], which uses 3 embedding vectors for each image.

Table 7 presents results for face verification. Our model achieves the best accuracy among all models trained on CASIA-WebFace. Also note that here our method outperforms models using a wide range of training procedures. MFM [38] use a softmax classification loss. CASIA [40] use a combination of softmax loss and contrastive loss. N-pair [28] use a more costly loss function that is defined on all pairs in a batch. We also list a few other state-of-the-art results which are not comparable purely for reference. DeepID2 [30] and DeepID3 [31] use 25 networks on 25 face regions based on positions of facial landmarks. When trained using only one network, their performance degrades significantly. Other models such as FaceNet [25] and DeepFace [33] are trained on huge private datasets.

Model	# training images	Accuracy (%)	Embed. dim.	# Nets
FaceNet [25]	200M	**99.63**	128	1
DeepFace [33]	4.4M	97.35	4096	1
MultiBatch [32]	2.6M	98.20	128	1
VGG [23]	2.6M	99.13	1024	1
DeepID2 [30]	203K	95.43	160	1
DeepID2 [30]	203K	99.15	160	25
DeepID3 [31]	300K	99.53	600	25
CASIA [40]	494k	97.30	320	1
MFM [38]	494k	98.13	256	1
N-pairs [28]	494k	98.33	320	1
Margin	494k	98.20	**128**	1
Margin	494k	**98.37**	256	1

Table 7: Face verification accuracy on LFW. We directly compare to results trained on CASIA-WebFace, shown in the lower part of the table. Methods shown in the upper part use either more or proprietary data, and are listed purely for reference.

Overall, our model achieves the best results on all datasets among all compared methods. Notably, our method uses the simplest loss function among all — a simple variant of contrastive loss.

6. Conclusion

We demonstrated that sampling matters as much or more than loss functions in deep embedding learning. This should not come as a surprise, since the implicitly defined loss function is (quite obviously) a sample weighted object.

Our new distance weighted sampling yields a performance improvement for multiple loss functions. In addition, we analyze and provide a simple margin-based loss that relaxes unnecessary constraints from traditional contrastive loss and enjoys the flexibility of the triplet loss. We show that distance weighted sampling and the margin based loss significantly outperform all other loss functions.

Acknowledgment

We would like to thank Manzil Zaheer for helpful discussions. This work was supported in part by Berkeley DeepDrive, and an equipment grant from Nvidia.

References

[1] The sphere game in n dimensions. http://faculty.madisoncollege.edu/alehnen/sphere/hypers.htm. Accessed: 2017-02-22. 3

[2] B. Amos, B. Ludwiczuk, and M. Satyanarayanan. Openface: A general-purpose face recognition library with mobile applications. Technical report, CMU, 2016. 3

[3] S. Bell and K. Bala. Learning visual similarity for product design with convolutional neural networks. *ACM TOG*, 2015. 1, 2, 7

[4] J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, and R. Shah. Signature verification using a "siamese" time delay neural network. *IJPRAI*, 1993. 1, 2

[5] M. Bucher, S. Herbin, and F. Jurie. Improving semantic embedding consistency by metric learning for zero-shot classiffication. In *ECCV*, 2016. 1

[6] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, 2005. 1, 2

[7] P. Comon. Independent component analysis, a new concept? *Signal processing*, 1994. 2

[8] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 1995. 5

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 6

[10] M. Hadi Kiapour, X. Han, S. Lazebnik, A. C. Berg, and T. L. Berg. Where to buy it: Matching street clothing photos in online shops. In *ICCV*, 2015. 1

[11] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR*, 2006. 1, 2, 5, 6

[12] E. Hazan, K. Levy, and S. Shalev-Shwartz. Beyond convexity: Stochastic quasi-convex optimization. In *NIPS*, 2015. 4

[13] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, 2016. 6

[14] R. Herbrich, T. Graepel, and K. Obermayer. Support vector learning for ordinal regression. In *ICANN*, 1999. 2

[15] C. Huang, C. C. Loy, and X. Tang. Local similarity-aware deep feature embedding. In *NIPS*, 2016. 2, 7, 8

[16] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical report, UMass Amherst, 2007. 1, 6

[17] T. Joachims. Optimizing search engines using clickthrough data. In *KDD*, 2002. 4

[18] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[19] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3d object representations for fine-grained categorization. In *Workshop on 3D Representation and Recognition, at ICCV*, 2013. 5, 8

[20] K. Y. Levy. The power of normalization: Faster evasion of saddle points. *arXiv preprint arXiv:1611.04831*, 2016. 4

[21] T. Moon, A. Smola, Y. Chang, and Z. Zheng. IntervalRank: Isotonic regression with listwise and pairwise constraints. In *WSDM*, 2010. 2, 5

[22] H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese. Deep metric learning via lifted structured feature embedding. In *CVPR*, 2016. 1, 2, 5, 7, 8

[23] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *BMVC*, 2015. 1, 2, 3, 6, 8

[24] B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural computation*, 2000. 5

[25] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 2015. 1, 2, 3, 4, 5, 6, 8

[26] M. Schultz and T. Joachims. Learning a distance metric from relative comparisons. In *NIPS*, 2003. 2

[27] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *ICCV*, 2015. 2

[28] K. Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *NIPS*, 2016. 1, 2, 7, 8

[29] H. O. Song, S. Jegelka, V. Rathod, and K. Murphy. Learnable structured clustering framework for deep metric learning. *arXiv preprint arXiv:1612.01213*, 2016. 1, 2, 7, 8

[30] Y. Sun, Y. Chen, X. Wang, and X. Tang. Deep learning face representation by joint identification-verification. In *NIPS*, 2014. 8

[31] Y. Sun, D. Liang, X. Wang, and X. Tang. Deepid3: Face recognition with very deep neural networks. *arXiv preprint arXiv:1502.00873*, 2015. 8

[32] O. Tadmor, T. Rosenwein, S. Shalev-Shwartz, Y. Wexler, and A. Shashua. Learning a metric embedding for face recognition using the multibatch method. In *NIPS*, 2016. 8

[33] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *CVPR*, 2014. 8

[34] E. Ustinova and V. Lempitsky. Learning deep embeddings with histogram loss. In *NIPS*, 2016. 1, 2, 7, 8

[35] M. Weimer, A. Karatzoglou, and A. Smola. CoFiRank: Collaborative filtering for ranking. https://github.com/markusweimer/cofirank, 2009. 2

[36] K. Q. Weinberger and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. *JMLR*, 2009. 2

[37] 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. 5, 8

[38] X. Wu, R. He, and Z. Sun. A lightened cnn for deep face representation. In *CVPR*, 2015. 8

[39] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning with application to clustering with side-information. In *NIPS*, 2002. 2

[40] D. Yi, Z. Lei, S. Liao, and S. Z. Li. Learning face representation from scratch. *arXiv preprint arXiv:1411.7923*, 2014. 6, 8

[41] Y. Yuan, K. Yang, and C. Zhang. Hard-aware deeply cascaded embedding. *arXiv preprint arXiv:1611.05720*, 2016. 1, 2, 8

[42] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 2016. 6

[43] P. Zhao and T. Zhang. Stochastic optimization with importance sampling for regularized loss minimization. In *ICML*, 2015. 2

[44] Z. Zheng, H. Zha, and G. Sun. Query-level learning to rank using isotonic regression. In *Allerton*, 2008. 5

[45] B. Zhuang, G. Lin, C. Shen, and I. Reid. Fast training of triplet-based deep binary embedding networks. In *CVPR*, 2016. 1

Appendix A. Empirical pairwise-distance distributions

To better understand the effects of distance weighted sampling during training, we analyze our learned embeddings. Specifically, we compute empirical pairwise distance distributions for negative pairs based on the embeddings of testing images. Figure 7 presents the results on Stanford Online Product dataset. We see that after the first epoch, the distribution already forms a bell shape, and in later epochs, it gradually concentrates. This justifies our motivation of using distance weighted sampling so that examples from all distances have a chance to be sampled.

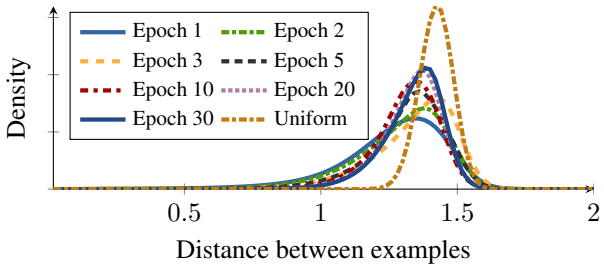


Figure 7: Empirical pairwise-distance distributions for negative pairs. They roughly follow a bell-shaped curve.

Appendix B. Stability analysis

Here we measure the stability of different loss functions when using different batch construction. Specifically, we change the number of images m per class in a batch and see how it impacts the solutions. For this purpose, we experiment with face verification and use the optimal verification boundary on the validation set as a summary of the solution. The results are summarized in Figure 8. We see that the triplet loss converges to different solutions when using different batch constructions. In addition, we observe large fluctuations in the early stage, indicating unstable training. On the other hand, the margin based loss is robust, it always converges to the roughly the same geometry.

Appendix C. Ablation study for batch size

We analyze the sensitivity of our approach with respect to batch sizes. Table 8 presents the results. We see that distance weighted sampling consistently outperforms other sampling strategies, and margin based loss consistently outperforms triplet loss.

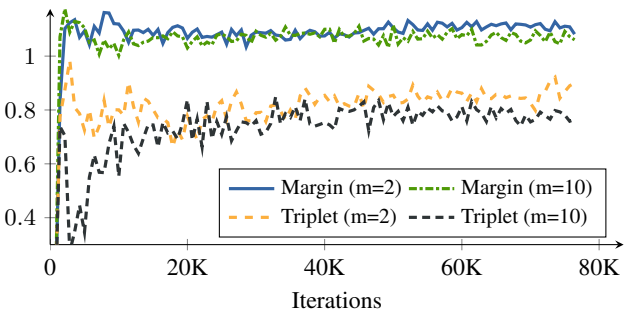


Figure 8: Optimal validation threshold for the LFW dataset. Triplet loss with different sampling strategies converges to different solutions. In addition, it has large fluctuations in the early stage, indicating unstable training. Margin based loss always converges stably to the same solution.

Loss, batch size	@1	@10	@100	@1000
Triplet ℓ_2, 40				
Semihard	44.3	63.7	79.7	92.2
Distance weighted	52.9	70.9	83.9	94.0
Triplet ℓ_2, 80				
Semihard	47.4	67.5	83.1	93.6
Distance weighted	54.5	72.0	85.4	94.4
Triplet ℓ_2, 120				
Semihard	48.8	67.7	82.7	93.3
Distance weighted	54.7	72.7	85.9	94.6
Margin, 40				
Random	41.9	60.2	76.3	89.6
Semihard	60.7	75.3	85.9	94.1
Distance weighted	61.1	75.8	86.5	94.2
Margin, 80				
Random	37.5	56.3	73.8	88.3
Semihard	61.0	74.6	85.3	93.6
Distance weighted	61.7	75.5	86.0	94.0
Margin, 120				
Random	37.7	56.6	73.7	88.3
Semihard	59.6	73.7	84.4	93.2
Distance weighted	60.5	74.7	85.5	93.8

Table 8: Recall@k evaluated on Stanford Online Products.