Proxy Anchor Loss for Deep Metric Learning

Sungyeon Kim Dongwon Kim Minsu Cho Suha Kwak
POSTECH, Pohang, Korea
{tjddus9597, kdwon, mscho, suha.kwak}@postech.ac.kr

Abstract

Existing metric learning losses can be categorized into two classes: pair-based and proxy-based losses. The former class can leverage fine-grained semantic relations between data points, but slows convergence in general due to its high training complexity. In contrast, the latter class enables fast and reliable convergence, but cannot consider the rich data-to-data relations. This paper presents a new proxy-based loss that takes advantages of both pair- and proxy-based methods and overcomes their limitations. Thanks to the use of proxies, our loss boosts the speed of convergence and is robust against noisy labels and outliers. At the same time, it allows embedding vectors of data to interact with each other through its gradients to exploit data-to-data relations. Our method is evaluated on four public benchmarks, where a standard network trained with our loss achieves state-of-the-art performance and most quickly converges.

1. Introduction

Learning a semantic distance metric has been a crucial step for many applications such as content-based image retrieval [14, 21, 27, 29], face verification [18, 25], person re-identification [3, 38], few-shot learning [24, 26, 30], and representation learning [14, 33, 41]. Following their great success in visual recognition, deep neural networks have been employed recently for metric learning. The networks are trained to project data onto an embedding space in which semantically similar data (*e.g.*, images of the same class) are closely grouped together. Such a quality of the embedding space is given mainly by loss functions used for training the networks, and most of the losses are categorized into two classes: *pair-based* and *proxy-based*.

The pair-based losses are built upon pairwise distances between data in the embedding space. A seminal example is Contrastive loss [4, 9], which aims to minimize the distance between a pair of data if their class labels are identical and to separate them otherwise. Recent pair-based losses consider a group of pairwise distances to handle relations between more than two data [14, 25, 27, 29, 32, 34, 35, 39].

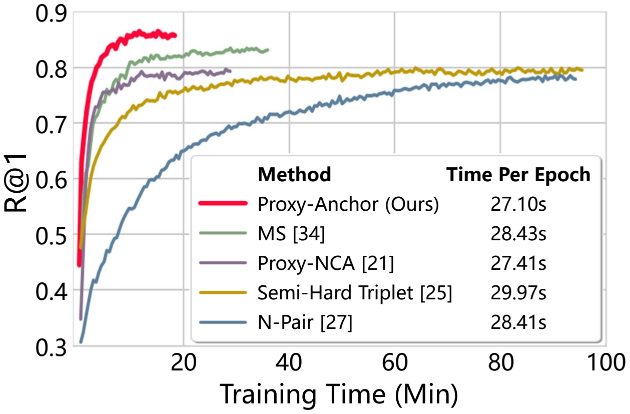


Figure 1. Accuracy in Recall@1 versus training time on the Cars-196 [17] dataset. Note that all methods were trained with batch size of 150 on a single Titan Xp GPU. Our loss enables to achieve the highest accuracy, and converge faster than the baselines in terms of both the number of epochs and the actual training time.

These losses provide rich supervisory signals for training embedding networks by comparing data to data and examining fine-grained relations between them, *i.e.*, *data-to-data relations*. However, since they take a tuple of data as a unit input, the losses cause prohibitively high training complexity[1], $O(M^2)$ or $O(M^3)$ where M is the number of training data, thus slow convergence. Furthermore, some tuples do not contribute to training or even degrade the quality of the learned embedding space. To resolve these issues, learning with the pair-based losses often entails tuple sampling techniques [10, 25, 37, 40], which however have to be tuned by hand and may increase the risk of overfitting.

The proxy-based losses resolve the above complexity issue by introducing *proxies* [1, 21, 23]. A proxy is a representative of a subset of training data and learned as a part of the network parameters. Existing losses in this category consider each data point as an anchor, associate it with proxies instead of other images, and encourage the anchor to be close to proxies of the same class and far apart from those of different classes. Proxy-based losses reduce the training

[1] The training complexity indicates the amount of computation required to address the entire training dataset [1, 6, 10, 23, 35].

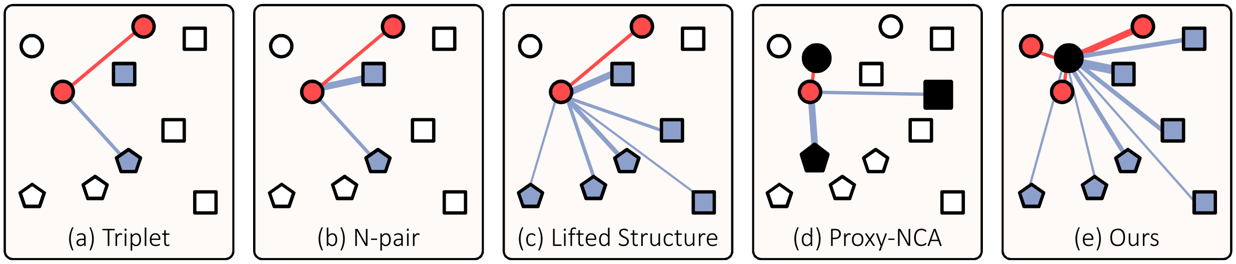


Figure 2. Comparison between popular metric learning losses and ours. Small nodes are embedding vectors of data in a batch, and black ones indicate proxies; their different shapes represent distinct classes. The associations defined by the losses are expressed by edges, and thicker edges get larger gradients. Also, embedding vectors associated with the anchor are colored in red if they are of the same class of the anchor (*i.e.*, positive) and in blue otherwise (*i.e.*, negative). (a) Triplet loss [25, 32] associates each anchor with a positive and a negative data point without considering their hardness. (b) N-pair loss [27] and (c) Lifted Structure loss [29] reflect hardness of data, but do not utilize all data in the batch. (d) Proxy-NCA loss [21] cannot exploit data-to-data relations since it associates each data point only with proxies. (e) Our loss handles entire data in the batch, and associates them with each proxy with consideration of their relative hardness determined by data-to-data relations. See the text for more details.

complexity and enable faster convergence since the number of proxies is substantially smaller than that of training data in general. Further, these losses tend to be more robust against label noises and outliers. However, since they associate each data point only with proxies, proxy-based losses can leverage only *data-to-proxy relations*, which are impoverished compared to the rich data-to-data relations available for pair-based losses.

In this paper, we propose a novel proxy-based loss called Proxy-Anchor loss, which takes good points of both proxy-based and pair-based losses while correcting their defects. Unlike the existing proxy-based losses, the proposed loss utilizes each proxy as an anchor and associates it with all data in a batch. Specifically, for each proxy, the loss aims to pull data of the same class close to the proxy and to push others away in the embedding space. Due to the use of proxies, our loss boosts the speed of convergence with no hyperparameter for tuple sampling, and is robust against noisy labels and outliers. At the same time, it can take data-to-data relations into account like pair-based losses; this property is given by associating all data in a batch with each proxy so that the gradients with respect to a data point are weighted by its relative proximity to the proxy (*i.e.*, relative hardness) affected by the other data in the batch. Thanks to the above advantages, a standard embedding network trained with our loss achieves state-of-the-art accuracy and most quickly converges as shown in Figure 1. The contribution of this paper is three-fold:

- We propose a novel metric learning loss that takes advantages of both pair-based and proxy-based methods; it leverages rich data-to-data relations and enables fast and reliable convergence.
- A standard embedding network trained with our loss achieves state-of-the-art performance on the four public benchmarks for metric learning [17, 19, 29, 36].
- Our loss speeds up convergence greatly without careful data sampling; its convergence is even faster than those of Proxy-NCA [21] and Multi-Similarity loss [34].

2. Related Work

In this section, we categorize metric learning losses into two classes, pair-based and proxy-based losses, then review relevant methods for each category.

2.1. Pair-based Losses

Contrastive loss [2, 4, 9] and Triplet loss [25, 32] are seminal examples of loss functions for deep metric learning. Contrastive loss takes a pair of embedding vectors as input, and aims to pull them together if they are of the same class and push them apart otherwise. Triplet loss considers a data point as an anchor, associates it with a positive and a negative data point, and constrains the distance of the anchor-positive pair to be smaller than that of the anchor-negative pair in the embedding space as illustrated in Figure 2(a).

Recent pair-based losses aim to leverage higher order relations between data and reflect their hardness for further enhancement. As generalizations of Triplet loss, N-pair loss [27] and Lifted Structure loss [29] associate an anchor with a single positive and multiple negative data points, and pull the positive to the anchor and push the negatives away from the anchor while considering their hardness. As shown in Figure 2(b) and 2(c), however, these losses do not utilize entire data in a batch since they sample the same number of data per negative class, thus may drop informative examples during training. In contrast, Ranked List loss [35] takes into account all positive and negative data in a batch and aims to separate the positive and negative sets. Multi-Similarity loss [34] also considers every pair of data in a batch, and assigns a weight to each pair according to three complementary types of similarity to focus more on useful pairs for improving performance and convergence speed.

Pair-based losses enjoy rich and fine-grained data-to-data relations as they examine tuples (*i.e*., data pairs or their combinations) during training. However, since the number of tuples increases polynomially with the number of training data, their training complexity is prohibitively high and convergence is slow. In addition, a large amount of tuples are not effective and sometimes even degrade the quality of the learned embedding space [25, 37]. To address this issue, most pair-based losses entail tuple sampling techniques [10, 25, 37, 40] to select and utilize tuples that will contribute to training. However, these techniques involve hyperparameters that have to be tuned carefully, and may increase the risk of overfitting since they rely mostly on local pairwise relations within a batch. Another way to alleviating the complexity issue is to assign larger weights to more useful pairs during training as in [34], which however also incorporates a sampling technique.

Our loss resolves this complexity issue by adopting proxies, which enables faster and more reliable convergence compared to pair-based losses. Furthermore, it demands no additional hyperparameter for tuple sampling.

2.2. Proxy-based Losses

Proxy-based metric learning is a relatively new approach that can address the complexity issue of the pair-based losses. A proxy means a representative of a subset of training data and is estimated as a part of the embedding network parameters. The common idea of the methods in this category is to infer a small set of proxies that capture the global structure of an embedding space and relate each data point with the proxies instead of the other data points during training. Since the number of proxies is significantly smaller than that of training data, the training complexity can be reduced substantially.

The first proxy-based loss is Proxy-NCA [21], which is an approximation of Neighborhood Component Analysis (NCA) [8] using proxies. In its standard setting, Proxy-NCA loss assigns a single proxy for each class, associates a data point with proxies, and encourages the positive pair to be close and negative pairs to be far apart, as illustrated in Figure 2(d). SoftTriple loss [23], an extension of SoftMax loss for classification, is similar to Proxy-NCA yet assigns multiple proxies to each class to reflect intra-class variance. Manifold Proxy loss [1] is an extension of N-pair loss using proxies, and improves the performance by adopting a manifold-aware distance instead of Euclidean distance to measure the semantic distance in the embedding space.

Using proxies in these losses helps improve training convergence greatly, but has an inherent limitation as a side effect: Since each data point is associated only with proxies, the rich data-to-data relations that are available for the pair-based methods are not accessible anymore. Our loss can overcome this limitation since its gradients reflect relative hardness of data and allow their embedding vectors to interact with each other during training.

3. Our Method

We propose a new metric learning loss called Proxy-Anchor loss to overcome the inherent limitations of the previous methods. The loss employs proxies that enable fast and reliable convergence as in proxy-based losses. Also, although it is built upon data-proxy relations, our loss can utilize data-to-data relations during training like pair-based losses since it enables embedding vectors of data points to be affected by each other through its gradients. This property of our loss improves the quality of the learned embedding space substantially.

In this section, we first review Proxy-NCA loss [21], a representative proxy-based loss, for comparison to our Proxy-Anchor loss. We then describe our Proxy-Anchor loss in detail and analyze its training complexity.

3.1. Review of Proxy-NCA Loss

In the standard setting, Proxy-NCA loss [21] assigns a proxy to each class so that the number of proxies is the same with that of class labels. Given an input data point as an anchor, the proxy of the same class of the input is regarded as positive and the other proxies are negative. Let x denote the embedding vector of the input, p^+ be the positive proxy, and p^- be a negative proxy. The loss is then given by

$$\begin{aligned}\ell(X) &= \sum_{x\in X} -\log \frac{e^{s(x,p^+)}}{\sum\limits_{p^-\in P^-} e^{s(x,p^-)}} && (1)\\ &= \sum_{x\in X} \Big\{ -s(x,p^+) + \underset{p^-\in P^-}{\text{LSE}}\, s(x,p^-)\Big\}, && (2)\end{aligned}$$

where X is a batch of embedding vectors, P^- is the set of negative proxies, and $s(\cdot,\cdot)$ denotes the cosine similarity between two vectors. In addition, LSE in Eq. (2) means the Log-Sum-Exp function, a smooth approximation to the max function. The gradient of Proxy-NCA loss with respect to $s(x,p)$ is given by

$$\frac{\partial \ell(X)}{\partial s(x,p)} = \begin{cases} -1, & \text{if } p = p^+, \\ \dfrac{e^{s(x,p)}}{\sum\limits_{p^-\in P^-} e^{s(x,p^-)}}, & \text{otherwise.} \end{cases} \qquad (3)$$

Eq. (3) shows that minimizing the loss encourages x and p^+ to be close to each other, and x and p^- to be far away. In particular, x and p^+ are pulled together by the constant power, while x and p^- closer to each other (*i.e*., harder negative) are more strongly pushed away.

Proxy-NCA loss enables fast convergence thanks to its low training complexity, $O(MC)$ where M is the number of training data and C is that of classes, which is substantially lower than $O(M^2)$ or $O(M^3)$ of pair-based losses since $C \ll M$; refer to Section 3.3 for details. Also, proxies are robust against outliers and noisy labels since they are trained to represent groups of data. However, since the loss associates each embedding vector only with proxies, it cannot exploit fine-grained data-to-data relations. This drawback limits the capability of embedding networks trained with Proxy-NCA loss.

3.2. Proxy-Anchor Loss

Our Proxy-Anchor loss is designed to overcome the limitation of Proxy-NCA while keeping the low training complexity. The main idea is to take each proxy as an anchor and associate it with entire data in a batch, as illustrated in Figure 2(e), so that the data interact with each other through the proxy anchor during training. Our loss assigns a proxy for each class following the standard proxy assignment setting of Proxy-NCA, and is formulated as

$$\begin{aligned}\ell(X) =& \frac{1}{|P^+|}\sum_{p\in P^+}\log\left(1+\sum_{x\in X_p^+}e^{-\alpha(s(x,p)-\delta)}\right)\\ &+\frac{1}{|P|}\sum_{p\in P}\log\left(1+\sum_{x\in X_p^-}e^{\alpha(s(x,p)+\delta)}\right),\end{aligned} \tag{4}$$

where $\delta > 0$ is a margin, $\alpha > 0$ is a scaling factor, P indicates the set of all proxies, and P^+ denotes the set of positive proxies of data in the batch. Also, for each proxy p, a batch of embedding vectors X is divided into two sets: X_p^+, the set of positive embedding vectors of p, and $X_p^- = X - X_p^+$. The proposed loss can be rewritten in an easier-to-interpret form as

$$\begin{aligned}\ell(X) =& \frac{1}{|P^+|}\sum_{p\in P^+}\left[\text{Softplus}\left(\underset{x\in X_p^+}{\text{LSE}} -\alpha(s(x,p)-\delta)\right)\right]\\ &+\frac{1}{|P|}\sum_{p\in P}\left[\text{Softplus}\left(\underset{x\in X_p^-}{\text{LSE}}\,\alpha(s(x,p)+\delta)\right)\right],\end{aligned} \tag{5}$$

where $\text{Softplus}(z) = \log(1+e^z), \forall z \in \mathbb{R}$, and is a smooth approximation of ReLU.

How it works: Regarding Log-Sum-Exp as the max function, it is easy to notice that the loss aims to pull p and its most dissimilar positive example (*i.e.*, hardest positive example) together, and to push p and its most similar negative example (*i.e.*, hardest negative example) apart. Due to the nature of Log-Sum-Exp, the loss in practice pulls and pushes all embedding vectors in the batch, but with different degrees of strength that are determined by their relative hardness. This characteristic is demonstrated by the gradient of our loss with respect to $s(x,p)$, which is given by

$$\frac{\partial \ell(X)}{\partial s(x,p)} = \begin{cases} \dfrac{1}{|P^+|}\dfrac{-\alpha\, h_p^+(x)}{1+\sum_{x'\in X_p^+} h_p^+(x')}, & \forall x \in X_p^+,\\[2ex] \dfrac{1}{|P|}\dfrac{\alpha\, h_p^-(x)}{1+\sum_{x'\in X_p^-} h_p^-(x')}, & \forall x \in X_p^-,\end{cases} \tag{6}$$

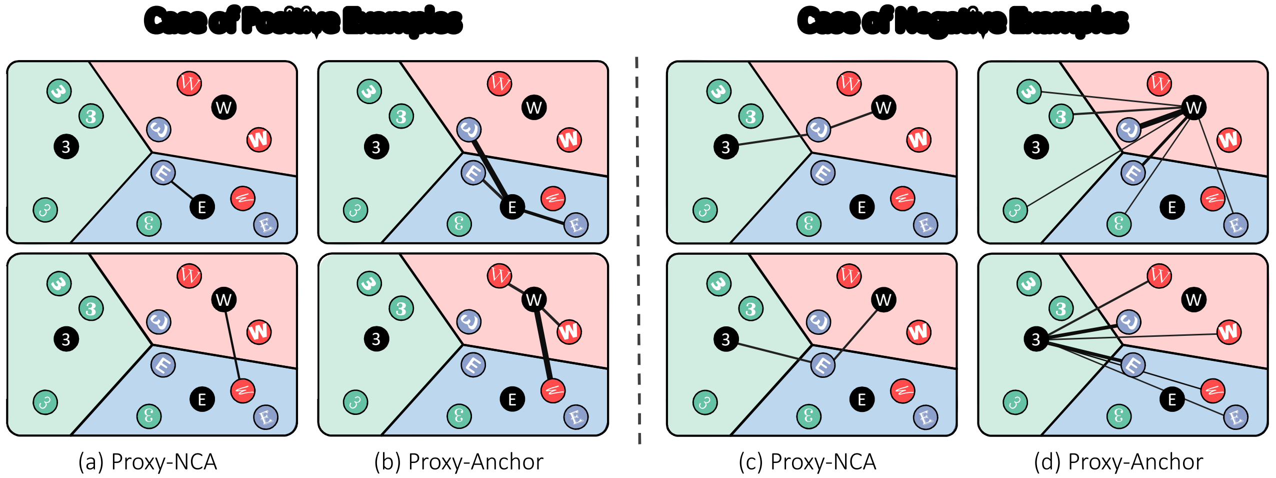
where $h_p^+(x) = e^{-\alpha(s(x,p)-\delta)}$ and $h_p^-(x) = e^{\alpha(s(x,p)+\delta)}$ are positive and negative hardness metrics for embedding vector x given proxy p, respectively; $h_p^+(x)$ is large when the positive embedding vector x is far from p, and $h_p^-(x)$ is large when the negative embedding vector x is close to p. The scaling parameter α and margin δ control the relative hardness of data points, and in consequence, determine how strongly pull or push their embedding vectors.

As shown in the above equations, the gradient for $s(x,p)$ is affected by not only x but also other embedding vectors in the batch; the gradient becomes larger when x is harder than the others. In this way, our loss enables embedding vectors in the batch to interact with each other and reflects their relative hardness through the gradients, which helps enhance the quality of the learned embedding space.

Comparison to Proxy-NCA: The key difference and advantage of Proxy-Anchor over Proxy-NCA is the active consideration of relative hardness based on data-to-data relations. This property enables Proxy-Anchor loss to provide richer supervisory signals to embedding networks during training. The gradients of the two losses demonstrate this clearly. In Proxy-NCA loss, the scale of the gradient is constant for every positive example and that of a negative example is calculated by taking only few proxies into account as shown in Eq. (3). In particular, the constant gradient scale for positive examples damages the flexibility and generalizability of embedding networks [37]. In contrast, Proxy-Anchor loss determines the scale of the gradient by taking relative hardness into consideration for both positive and negative examples as shown in Eq. (6). This feature of our loss allows the embedding network to consider data-to-data relations that are ignored in Proxy-NCA and observe much larger area of the embedding space during training than Proxy-NCA. Figure 3 illustrates these differences between the two losses in terms of handling the relative hardness of embedding vectors. In addition, unlike Proxy-Anchor loss, the margin imposed in our loss leads to intra-class compactness and inter-class separability, resulting in a more discriminative embedding space.

3.3. Training Complexity Analysis

Let M, C, B, and U denote the numbers of training samples, classes, batches per epoch, and proxies held by each



(a) Proxy-NCA (b) Proxy-Anchor (c) Proxy-NCA (d) Proxy-Anchor

Figure 3. Differences between Proxy-NCA and Proxy-Anchor in handling proxies and embedding vectors during training. Each proxy is colored in black and three different colors indicate distinct classes. The associations defined by the losses are expressed by edges, and thicker edges get larger gradients. (a) Gradients of Proxy-NCA loss with respect to positive examples have the same scale regardless of their hardness. (b) Proxy-Anchor loss dynamically determines gradient scales regarding relative hardness of all positive examples so as to pull harder positives more strongly. (c) In Proxy-NCA, each negative example is pushed only by a small number of proxies without considering the distribution of embedding vectors in fine details. (d) Proxy-Anchor loss considers the distribution of embedding vectors in more details as it has all negative examples affect each other in their gradients.

Type	Loss	Training Complexity
Proxy	Proxy-Anchor (Ours)	$O(MC)$
	Proxy-NCA [21]	$O(MC)$
	SoftTriple [23]	$O(MCU^2)$
Pair	Contrastive [2, 4, 9]	$O(M^2)$
	Triplet (Semi-Hard) [25]	$O(M^3/B^2)$
	Triplet (Smart) [10]	$O(M^2)$
	N-pair [27]	$O(M^3)$
	Lifted Structure [29]	$O(M^3)$

Table 1. Comparison of training complexities.

class, respectively. U is 1 thus ignored in most of proxy-based losses including ours, but is nontrivial for those managing multiple proxies per class such as SoftTriple loss [23].

Table 1 compares the training complexity of our loss with those of popular pair- and proxy-based losses. The complexity of our loss is $O(MC)$ since it compares every proxy with all positive or all negative examples in a batch. More specifically, in Eq. (4), the complexity of the first summation is $O(MC)$ and that of the second summation is also $O(MC)$, hence the total training complexity is $O(MC)$. The complexity of Proxy-NCA [21] is also $O(MC)$ since each data point is associated with one positive proxy and $C-1$ negative proxies as can be seen in Eq. (2). On the other hand, SoftTriple loss [23], a modification of SoftMax using multiple proxies per class, associates each data point with U positive proxies and $U(C-1)$ negative proxies. The total training complexity of this loss is thus $O(MCU^2)$. In conclusion, the complexity of our loss is the same with or even lower than that of other proxy-based losses.

The training complexity of pair-based losses is higher than that of proxy-based ones. Since Contrastive loss [2, 4, 9] takes a pair of data as input, its training complexity is $O(M^2)$. On the other hand, Triplet loss that examines triplets of data has complexity $O(M^3)$, which can be reduced by triplet mining strategies. For example, semi-hard mining [25] reduces the complexity to $O(M^3/B^2)$ by selecting negative pairs that are located within a neighborhood of anchor but sufficiently far from it. Similarly, Smart mining [10] lowers the complexity to $O(M^2)$ by sampling hard triplets using an approximated nearest neighbor index. However, even with these techniques, the training complexity of Triplet loss is still high. Like Triplet loss, N-pair loss [27] and Lifted Structure loss [29] that compare each positive pair of data to multiple negative pairs also have complexity $O(M^3)$. The training complexity of these losses becomes prohibitively high as the number of training data M increases, which slows down the speed of convergence as demonstrated in Figure 1.

4. Experiments

In this section, our method is evaluated and compared to current state-of-the-art on the four benchmark datasets for deep metric learning [17, 19, 29, 36]. We also investigate the effect of hyperparameters and embedding dimensionality of our loss to demonstrate its robustness.

Recall@K		CUB-200-2011				Cars-196			
		1	2	4	8	1	2	4	8
Clustering^{64} [28]	BN	48.2	61.4	71.8	81.9	58.1	70.6	80.3	87.8
Proxy-NCA^{64} [21]	BN	49.2	61.9	67.9	72.4	73.2	82.4	86.4	87.8
Smart Mining^{64} [10]	G	49.8	62.3	74.1	83.3	64.7	76.2	84.2	90.2
MS^{64} [34]	BN	57.4	69.8	80.0	87.8	77.3	85.3	90.5	94.2
SoftTriple^{64} [23]	BN	60.1	71.9	81.2	88.5	78.6	86.6	91.8	95.4
Proxy-Anchor^{64}	BN	**61.7**	**73.0**	**81.8**	**88.8**	**78.8**	**87.0**	**92.2**	**95.5**
Margin^{128} [37]	R50	63.6	74.4	83.1	90.0	79.6	86.5	91.9	95.1
HDC^{384} [40]	G	53.6	65.7	77.0	85.6	73.7	83.2	89.5	93.8
A-BIER^{512} [22]	G	57.5	68.7	78.3	86.2	82.0	89.0	93.2	96.1
ABE^{512} [15]	G	60.6	71.5	79.8	87.4	85.2	90.5	94.0	96.1
HTL^{512} [7]	BN	57.1	68.8	78.7	86.5	81.4	88.0	92.7	95.7
RLL-H^{512} [35]	BN	57.4	69.7	79.2	86.9	74.0	83.6	90.1	94.1
MS^{512} [34]	BN	65.7	77.0	86.3	91.2	84.1	90.4	94.0	96.5
SoftTriple^{512} [23]	BN	65.4	76.4	84.5	90.4	84.5	90.7	94.5	96.9
Proxy-Anchor^{512}	BN	**68.4**	**79.2**	**86.8**	**91.6**	**86.1**	**91.7**	**95.0**	**97.3**
†Contra+HORDE^{512} [13]	BN	66.3	76.7	84.7	90.6	83.9	90.3	94.1	96.3
†Proxy-Anchor^{512}	BN	**71.1**	**80.4**	**87.4**	**92.5**	**88.3**	**93.1**	**95.7**	**97.5**

Table 2. Recall@K (%) on the CUB-200-2011 and Cars-196 datasets. Superscripts denote embedding sizes and † indicates models using larger input images. Backbone networks of the models are denoted by abbreviations: G–GoogleNet [31], BN–Inception with batch normalization [12], R50–ResNet50 [11].

4.1. Datasets

We employ CUB-200-2011 [36], Cars-196 [17], Stanford Online Product (SOP) [29] and In-shop Clothes Retrieval (In-Shop) [19] datasets for evaluation. For CUB-200-2011, we use 5,864 images of its first 100 classes for training and 5,924 images of the other classes for testing. For Cars-196, 8,054 images of its first 98 classes are used for training and 8,131 images of the other classes are kept for testing. For SOP, we follow the standard dataset split in [29] using 59,551 images of 11,318 classes for training and 60,502 images of the rest classes for testing. Also for In-Shop, we follow the setting in [19] using 25,882 images of the first 3,997 classes for training and 28,760 images of the other classes for testing; the test set is further partitioned into a query set with 14,218 images of 3,985 classes and a gallery set with 12,612 images of 3,985 classes.

4.2. Implementation Details

Embedding network: For a fair comparison to previous work, the Inception network with batch normalization [12] pre-trained for ImageNet classification [5] is adopted as our embedding network. We change the size of its last fully connected layer according to the dimensionality of embedding vectors, and L_2-normalize the final output.

Training: In every experiment, we employ AdamW optimizer [20], which has the same update step of Adam [16] yet decays the weight separately. Our model is trained for 40 epochs with initial learning rate 10^{-4} on the CUB-200-2011 and Cars-196, and for 60 epochs with initial learning rate $6 \cdot 10^{-4}$ on the SOP and In-shop. The learning rate for proxies is scaled up 100 times for faster convergence. Input batches are randomly sampled during training.

Proxy setting: We assign a single proxy for each semantic class following Proxy-NCA [21]. The proxies are initialized using a normal distribution to ensure that they are uniformly distributed on the unit hypersphere.

Image setting: Input images are augmented by random cropping and horizontal flipping during training while they are center-cropped in testing. The default size of cropped images is 224×224 as in most of previous work, but for comparison to HORDE [13], we also implement models trained and tested with 256×256 cropped images.

Hyperparameter setting: α and δ in Eq. (4) is set to 32 and 10^{-1}, respectively, for all experiments.

4.3. Comparison to Other Methods

We demonstrate the superiority of our Proxy-Anchor loss quantitatively by evaluating its image retrieval performance on the four benchmark datasets. For a fair comparison to previous work, the accuracy of our model is measured in three different settings: 64/128 embedding dimension with the default image size (224×224), 512 embedding dimension with the default image size, and 512 embedding dimension with the larger image size (256×256).

Results on the CUB-200-2011 and Cars-196 datasets are summarized in Table 2. Our model outperforms all the previous arts including ensemble methods [15, 22] in all the three settings. In particular, on the challenging CUB-200-2011 dataset, it improves the previous best score by a large margin, 2.7% in Recall@1. As reported in Table 3,

Recall@K	1	10	100	1000
Clustering^{64} [28]	67.0	83.7	93.2	-
Proxy-NCA^{64} [21]	73.7	-	-	-
MS^{64} [34]	74.1	87.8	94.7	**98.2**
SoftTriple^{64} [23]	76.3	**89.1**	**95.3**	-
Proxy-Anchor^{64}	**76.5**	89.0	95.1	**98.2**
Margin^{128} [37]	72.7	86.2	93.8	98.0
HDC^{384} [40]	69.5	84.4	92.8	97.7
A-BIER^{512} [22]	74.2	86.9	94.0	97.8
ABE^{512} [15]	76.3	88.4	94.8	98.2
HTL^{512} [7]	74.8	88.3	94.8	98.4
RLL-H^{512} [35]	76.1	89.1	95.4	-
MS^{512} [34]	78.2	90.5	96.0	**98.7**
SoftTriple^{512} [23]	78.3	90.3	95.9	-
Proxy-Anchor^{512}	**79.1**	**90.8**	**96.2**	**98.7**
†Contra+HORDE^{512} [13]	80.1	91.3	96.2	**98.7**
†Proxy-Anchor^{512}	**80.3**	**91.4**	**96.4**	**98.7**

Table 3. Recall@K (%) on the SOP. Superscripts denote embedding sizes and † indicates models using larger input images.

Recall@K	1	10	20	40
HDC^{384} [40]	62.1	84.9	89.0	92.3
HTL^{128} [7]	80.9	94.3	95.8	97.4
MS^{128} [34]	88.0	97.2	98.1	98.7
Proxy-Anchor^{128}	**90.8**	**97.9**	**98.5**	**99.0**
FashionNet^{4096} [19]	53.0	73.0	76.0	79.0
A-BIER^{512} [22]	83.1	95.1	96.9	97.8
ABE^{512} [15]	87.3	96.7	97.9	98.5
MS^{512} [34]	89.7	97.9	98.5	99.1
Proxy-Anchor^{512}	**91.5**	**98.1**	**98.8**	**99.1**
†Contra+HORDE^{512} [13]	90.4	97.8	98.4	98.9
†Proxy-Anchor^{512}	**92.6**	**98.3**	**98.9**	**99.3**

Table 4. Recall@K (%) on the In-Shop. Superscripts denote embedding sizes and † indicates models using larger input images.

our model also achieves state-of-the-art performance on the SOP dataset. It outperforms previous models in all the cases except for Recall@10 and Recall@100 with 64 dimensional embedding, but even in these cases it achieves the second best. Finally, on the In-Shop dataset, it attains the best scores in all the three settings as shown in Table 4.

For all the datasets, our model with the larger crop size and 512 dimensional embedding achieves the state-of-the-art performance. Also note that our model with the low embedding dimension often outperforms existing models with the high embedding dimension, which suggests that our loss allows to learn a more compact yet effective embedding space. Last, but not least, our loss boosts the convergence speed greatly as summarized in Figure 1.

4.4. Qualitative Results

To further demonstrate the superiority of our loss, we present qualitative retrieval results of our model on the four datasets. As can be seen in Figure 4, intra-class appearance variation is significantly large in these datasets in particular by pose variation and background clutter in the CUB200-2011, distinct object colors in the Cars-196, and view-point changes in the SOP and In-Shop datasets. Even with these challenges, the embedding network trained with our loss performs retrieval robustly.

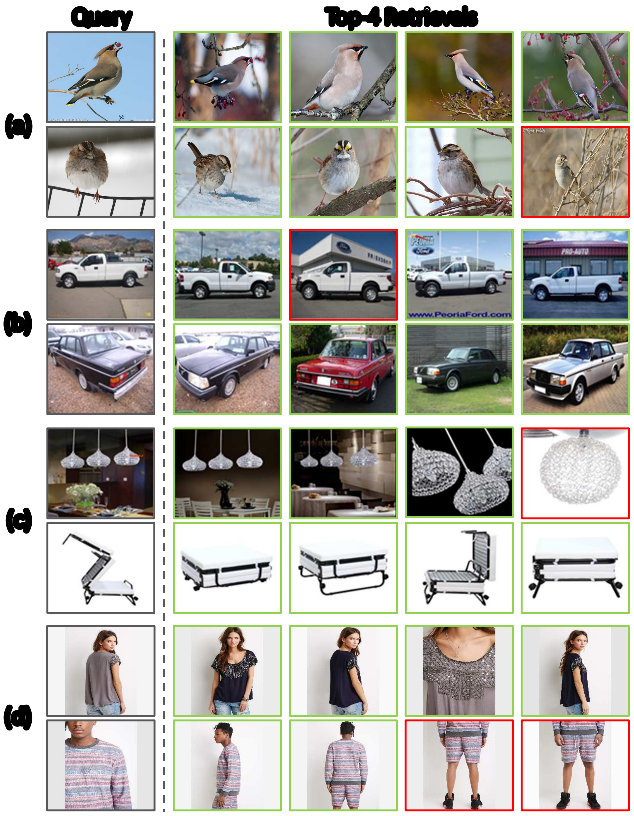


Figure 4. Qualitative results on the CUB-200-2011 (a), Cars-196 (b), SOP (c) and In-shop (d). For each query image (*leftmost*), top-4 retrievals are presented. The results with red boundary are failure cases, which are however substantially similar to their query images in terms of appearance.

4.5. Impact of Hyperparameters

Batch size: To investigate the effect of batch size on the performance of our loss, we examine Recall@1 of our loss while varying batch size on the four benchmark datasets. The result of the analysis is summarized in Table 5 and 6, where one can observe that larger batch sizes improve performance since our loss can consider a larger number of examples and their relations within each batch. On the other hand, performance is slightly reduced when the batch size is small since it is difficult to determine the relative hardness in this setting. On the datasets with a large number of images and classes, *i.e.*, SOP and In-shop, our loss needs to utilize more examples to fully leverage the relations be-

Batch size	Recall@1	
	CUB-200-2011	Cars-196
30	65.9	84.6
60	67.0	86.2
90	68.4	86.2
120	68.5	86.3
150	68.6	**86.4**
180	**69.0**	86.2

Table 5. Accuracy of our model in Recall@1 versus batch size on the CUB-200-2011 and Cars-196.

Batch size	Recall@1	
	SOP	In-shop
30	76.0	91.3
60	78.0	91.3
90	78.5	91.5
120	78.9	91.7
150	79.1	91.9
300	**79.3**	**92.0**
600	**79.3**	91.7

Table 6. Accuracy of our model in Recall@1 versus batch size on the SOP and In-shop.

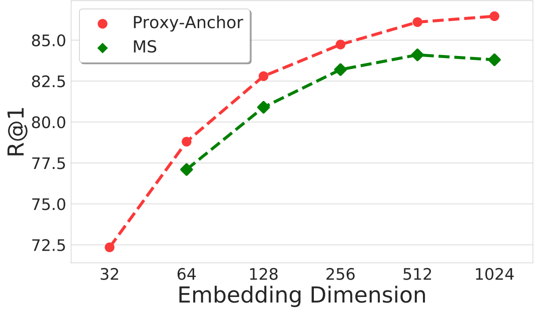


Figure 5. Accuracy in Recall@1 versus embedding dimension on the Cars-196.

tween data points. Our loss achieves the best performance when the batch size is equal to or larger than 300.

Embedding dimension: The dimension of embedding vectors is a crucial factor that controls the trade-off between speed and accuracy in image retrieval systems. We thus investigate the effect of embedding dimensions on the retrieval accuracy in our Proxy-Anchor loss. We test our loss with embedding dimensions varying from 64 to 1,024 following the experiment in [34], and further examine that with 32 embedding dimension. The result of analysis is quantified in Figure 5, in which the retrieval performance of our loss is compared with that of MS loss [34]. The performance of our loss is fairly stable when the dimension is equal to or larger than 128. Moreover, our loss outperforms MS loss in all embedding dimensions, and more importantly, its accuracy does not degrade even with the very high dimensional embedding unlike MS loss.

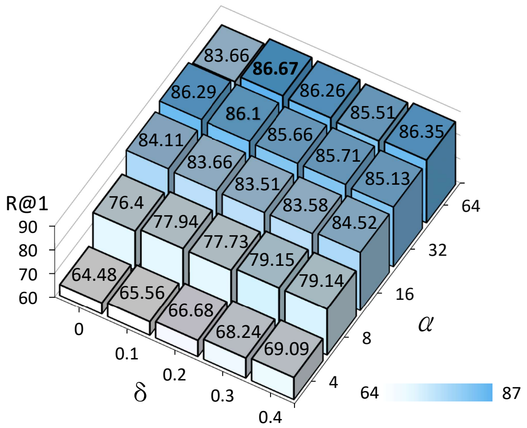


Figure 6. Accuracy in Recall@1 versus δ and α on the Cars-196.

α and δ of our loss: We also investigate the effect of the two hyperparameters α and δ of our loss on the Cars-196 dataset. The results of our analysis are summarized in Figure 6, in which we examine Recall@1 of Proxy-Anchor by varying the values of the hyperparameters $\alpha \in \{4, 8, 16, 32, 64\}$ and $\delta \in \{0, 0.1, 0.2, 0.3, 0.4\}$. The results suggest that when α is greater than 16, the accuracy of our model is high and stable, thus insensitive to the hyperparameter setting. Our loss outperforms current state-of-the-art with any α greater than 16. In addition, increasing δ improves performance although its effect is relatively small when α is large. Note that our hyperparameter setting reported in Section 4.2 is not the best, although it outperforms all existing methods on the dataset, as we did not tune the hyperparameters to optimize the test accuracy.

5. Conclusion

We have proposed a novel metric learning loss that takes advantages of both proxy- and pair-based losses. Like proxy-based losses, it enables fast and reliable convergence, and like pair-based losses, it can leverage rich data-to-data relations during training. As a result, our model has achieved state-of-the-art performance on the four public benchmark datasets, and at the same time, converged most quickly with no careful data sampling technique. In the future, we will explore extensions of our loss for deep hashing networks to improve its computational efficiency in testing as well as that in training.

Acknowledgement: This work was supported by IITP grant, Basic Science Research Program, and R&D program for Advanced Integrated-intelligence for IDentification through the NRF funded by the Ministry of Science, ICT (No.2019-0-01906 Artificial Intelligence Graduate School Program (POSTECH), NRF-2018R1C1B6001223, NRF-2018R1A5A1060031, NRF-2018M3E3A1057306, NRF-2017R1E1A1A01077999).

References

[1] Nicolas Aziere and Sinisa Todorovic. Ensemble deep manifold similarity learning using hard proxies. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 3

[2] Jane Bromley, Isabelle Guyon, Yann Lecun, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. In *Proc. Neural Information Processing Systems (NeurIPS)*, 1994. 2, 5

[3] Weihua Chen, Xiaotang Chen, Jianguo Zhang, and Kaiqi Huang. Beyond triplet loss: A deep quadruplet network for person re-identification. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1

[4] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. 1, 2, 5

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: a large-scale hierarchical image database. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 6

[6] Thanh-Toan Do, Toan Tran, Ian Reid, Vijay Kumar, Tuan Hoang, and Gustavo Carneiro. A theoretically sound upper bound on the triplet loss for improving the efficiency of deep distance metric learning. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1

[7] Weifeng Ge, Weilin Huang, Dengke Dong, and Matthew R. Scott. Deep metric learning with hierarchical triplet loss. In *Proc. European Conference on Computer Vision (ECCV)*, 2018. 6, 7

[8] Jacob Goldberger, Geoffrey E Hinton, Sam T Roweis, and Ruslan R Salakhutdinov. Neighbourhood components analysis. In *Proc. Neural Information Processing Systems (NeurIPS)*, 2005. 3

[9] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006. 1, 2, 5

[10] Ben Harwood, Vijay Kumar B G, Gustavo Carneiro, Ian Reid, and Tom Drummond. Smart mining for deep metric learning. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2017. 1, 3, 5, 6

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 6

[12] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. International Conference on Machine Learning (ICML)*, 2015. 6

[13] Pierre Jacob, David Picard, Aymeric Histace, and Edouard Klein. Metric learning with horde: High-order regularizer for deep embeddings. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2019. 6, 7

[14] Sungyeon Kim, Minkyo Seo, Ivan Laptev, Minsu Cho, and Suha Kwak. Deep metric learning beyond binary supervision. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1

[15] Wonsik Kim, Bhavya Goyal, Kunal Chawla, Jungmin Lee, and Keunjoo Kwon. Attention-based ensemble for deep metric learning. In *Proc. European Conference on Computer Vision (ECCV)*, 2018. 6, 7

[16] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. International Conference on Learning Representations (ICLR)*, 2015. 6

[17] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 554–561, 2013. 1, 2, 5, 6

[18] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1

[19] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 5, 6, 7

[20] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Proc. International Conference on Learning Representations (ICLR)*, 2019. 6

[21] Yair Movshovitz-Attias, Alexander Toshev, Thomas K Leung, Sergey Ioffe, and Saurabh Singh. No fuss distance metric learning using proxies. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2017. 1, 2, 3, 5, 6, 7

[22] Michael Opitz, Georg Waltner, Horst Possegger, and Horst Bischof. Deep metric learning with bier: Boosting independent embeddings robustly. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2018. 6, 7

[23] Qi Qian, Lei Shang, Baigui Sun, Juhua Hu, Hao Li, and Rong Jin. Softtriple loss: Deep metric learning without triplet sampling. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2019. 1, 3, 5, 6, 7

[24] Limeng Qiao, Yemin Shi, Jia Li, Yaowei Wang, Tiejun Huang, and Yonghong Tian. Transductive episodic-wise adaptive metric for few-shot learning. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2019. 1

[25] Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A unified embedding for face recognition and clustering. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 1, 2, 3, 5

[26] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087, 2017. 1

[27] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Proc. Neural Information Processing Systems (NeurIPS)*, 2016. 1, 2, 5

[28] Hyun Oh Song, Stefanie Jegelka, Vivek Rathod, and Kevin Murphy. Deep metric learning via facility location. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 6, 7

[29] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 2, 5, 6

[30] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proc. IEEE Confer-

ence on Computer Vision and Pattern Recognition (CVPR), 2018. 1

[31] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 6

[32] Jiang Wang, Yang Song, T. Leung, C. Rosenberg, Jingbin Wang, J. Philbin, Bo Chen, and Ying Wu. Learning fine-grained image similarity with deep ranking. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 1, 2

[33] Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2015. 1

[34] Xun Wang, Xintong Han, Weilin Huang, Dengke Dong, and Matthew R Scott. Multi-similarity loss with general pair weighting for deep metric learning. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2, 3, 6, 7, 8

[35] Xinshao Wang, Yang Hua, Elyor Kodirov, Guosheng Hu, Romain Garnier, and Neil M Robertson. Ranked list loss for deep metric learning. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2, 6, 7

[36] 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. 2, 5, 6

[37] Chao-Yuan Wu, R. Manmatha, Alexander J. Smola, and Philipp Krahenbuhl. Sampling matters in deep embedding learning. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2017. 1, 3, 4, 6, 7

[38] Tong Xiao, Shuang Li, Bochao Wang, Liang Lin, and Xiaogang Wang. Joint detection and identification feature learning for person search. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1

[39] Baosheng Yu and Dacheng Tao. Deep metric learning with tuplet margin loss. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2019. 1

[40] Yuhui Yuan, Kuiyuan Yang, and Chao Zhang. Hard-aware deeply cascaded embedding. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2017. 1, 3, 6, 7

[41] Sergey Zagoruyko and Nikos Komodakis. Learning to compare image patches via convolutional neural networks. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 1