Recall@k Surrogate Loss with Large Batches and Similarity Mixup

Yash Patel Giorgos Tolias Jiří Matas
Visual Recognition Group, Czech Technical University in Prague
{patelyas,toliageo,matas}@fel.cvut.cz

Abstract

This work focuses on learning deep visual representation models for retrieval by exploring the interplay between a new loss function, the batch size, and a new regularization approach. Direct optimization, by gradient descent, of an evaluation metric, is not possible when it is non-differentiable, which is the case for recall in retrieval. A differentiable surrogate loss for the recall is proposed in this work. Using an implementation that sidesteps the hardware constraints of the GPU memory, the method trains with a very large batch size, which is essential for metrics computed on the entire retrieval database. It is assisted by an efficient mixup regularization approach that operates on pairwise scalar similarities and virtually increases the batch size further. The suggested method achieves state-of-the-art performance in several image retrieval benchmarks when used for deep metric learning. For instance-level recognition, the method outperforms similar approaches that train using an approximation of average precision.

1. Introduction

Minimization of a loss that is a function of the test-time evaluation metric has shown to be beneficial in deep learning for numerous computer vision and natural language processing tasks. Examples include intersection-over-union as a loss that boosts performance for object detection [48, 70] and semantic segmentation [37], and structural similarity [34], peak signal-to-noise ratio [4] and perceptual [40] as reconstruction losses for image compression that give better results according to the respective evaluation metrics.

Training deep networks via gradient descent on the evaluation metric is not possible when the metric is non-differentiable. Deep learning methods resort to a proxy loss, a differentiable function, as a workaround, which empirically leads to a reasonable performance but may not align well with the evaluation metric. Examples exist in object detection [70], scene text recognition [42, 43], machine translation [3] and image retrieval [6, 41].

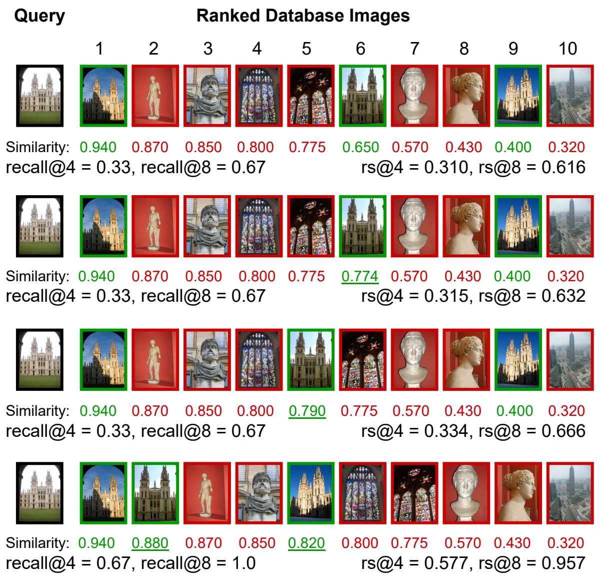
Query
Ranked Database Images
1 2 3 4 5 6 7 8 9 10
Similarity: 0.940 0.870 0.850 0.800 0.775 0.650 0.570 0.430 0.400 0.320
recall@4 = 0.33, recall@8 = 0.67
rs@4 = 0.310, rs@8 = 0.616
Similarity: 0.940 0.870 0.850 0.800 0.775 0.774 0.570 0.430 0.400 0.320
recall@4 = 0.33, recall@8 = 0.67
rs@4 = 0.315, rs@8 = 0.632
Similarity: 0.940 0.870 0.850 0.800 0.790 0.775 0.570 0.430 0.400 0.320
recall@4 = 0.33, recall@8 = 0.67
rs@4 = 0.334, rs@8 = 0.666
Similarity: 0.940 0.880 0.870 0.850 0.820 0.800 0.775 0.570 0.430 0.320
recall@4 = 0.67, recall@8 = 1.0
rs@4 = 0.577, rs@8 = 0.957

Figure 1. A comparison between recall@k and rs@k, the proposed differentiable recall@k surrogate. Examples show a query, the ranked database images sorted according to the similarity and the corresponding values for recall@k and rs@k and their dependence on similarity score change. Note that the values of recall@k and rs@k are close. Changes to similarity and ranking in some cases may not affect the original recall@k but can affect the surrogate, with the latter having a more significant impact than the former. Similarity values of all negatives are fixed for ease of understanding. The similarity values of the positives that were changed in rows 2, 3 and 4 are underlined.

This paper deals with the training of image retrieval posed as deep metric learning and Euclidean search in the learned image embedding space. It is the task of ranking all database examples according to the relevance to a query, which is of vital importance for many applications. The standard evaluation metrics are precision and recall in the top retrieved results and the mean Average Precision (mAP). These metrics are standard in information retrieval, they reflect the quality of the retrieved results and allow for flexibility to focus either on the few top results or the whole ranked list of examples, respectively. Recall at top-k retrieved results, denoted by *recall@k* in the following, is the primary focus of this work.

The problem related to the optimization of non-differentiable evaluation metrics applies to recall@k as well. Estimating the position of positive images in the list of retrieved results and counting how many positives appear inside a short-list of a fixed size involves non-differentiable operations. Note that methods for training on non-differentiable losses, such as actor-critic [3] and learning surrogates [42] are not directly applicable to recall@k. This is due to the fact that these methods are limited to decomposable functions, where a per-example performance measure is available. Such an attempt is made by Engilberge *et al*. [13], where an LSTM learns sorting-based metrics, but is not adapted in consequent work due to slow training. As an alternative, deep metric learning approaches for image retrieval often use ranking proxy losses, termed pairwise losses. In the embedding space, loss functions such as contrastive [18], triplet [53], and margin [69] pull the examples from the same class closer to one another and push the examples from a different class away. These losses are hand-crafted to reflect the objectives of the retrieval task and, consequently, the evaluation metric. The loss value depends on the image-to-image similarity for image pairs or triplets and does not take into account the whole ranked list of examples. Changes in the similarity value without any change in the overall ranking alter the loss value indicate that they are not well correlated with ranking [6]. Recent methods focus on optimizing Average Precision (AP) and use a surrogate function as a loss [6, 7, 19, 47, 49]. A surrogate of an evaluation metric is a function that approximates it in a differentiable manner.

The proposed method attains state-of-the-art results for 4 fine-grained retrieval datasets, namely iNaturalist [61], VehicleID [61], SOP [39] and Cars196 [27], and 2 instance-level retrieval datasets, namely Revisited Oxford and Paris [45]. This is accomplished by the demonstrated synergy between the three following elements. First, a new loss that is proposed as a surrogate of an established retrieval evaluation metric, namely recall at top k, and is experimentally shown to consistently outperform existing competitors. A comparison between the evaluation metric and the proposed loss is shown in Figure 1. Second, the use of a very large batch size, in the order of several thousand large resolution images on a single GPU. This is inspired by the instance-level retrieval literature [47] and is introduced for the first time in the context of fine-grained categorization. In a recent work of verifying prior results in deep metric learning for fine-grained categorization [36] the batch-size is considered fixed to a single and small value among a large set of comparisons for different losses; in this work we reach batch-sizes that are two orders of magnitude larger than in the work of Musgrave *et al*. [36]. The third elements is the proposed mixup regularization technique that is computationally efficient and that virtually enlarges the batch. Its efficiency is obtained by operating on the very last stage of similarity estimation, *i.e*. scalar similarities are mixed, while its applicability goes beyond the combination with the proposed loss in this work. The proposed loss is used for training widely used ResNet architectures [20] but also recent vision-transformers (ViT) [10]. The superiority of this loss compared to existing losses is demonstrated with both architectures, while with ViT-B/16 top results are achieved at lower throughput than with ResNet.

2. Related work

In this section, the related work is reviewed for two different families of deep metric learning approaches regarding the type of loss that is optimized, namely classification losses and pairwise losses. Given an embedding network that maps input images to a high dimensional space, in the former, the loss is a function of the embedding and the corresponding category label of a single image, while in the latter, the loss is a function of the distance, or similarity, between two embeddings and the corresponding pairwise label. Prior work for mixup [72] techniques related to embedding learning is reviewed too.

Classification losses. The work of Zhai and Wu [71] supports that the standard classification loss, *i.e*. cross-entropy (CE) loss is a strong approach for deep metric learning. Their finding is supported by the use of layer normalization and class-balanced sampling. In the domain of metric learning for faces, several different classification losses are proposed, such as SphereFace [30], CosFace [64] and ArcFace [8], where contributions are in the spirit of large margin classification. Despite the specificity of the domain, such losses are applicable beyond faces. Another variant is the Neighborhood Component Analysis (NCA) loss that is used in the work of Movshovitz-Attias *et al*. [35], which is later improved [58] by temperature-based scaling and faster update of the class prototype vectors, also called proxies in their work. The restriction of a single prototype vector per class is dropped by Qian *et al*. [44] who stores multiple representatives per category.

Classification losses, in contrast to pairwise losses, perform the optimization independently per image. An exception is the work of Elezi *et al*. [12] where a similarity propagation module captures group interactions within the batch. Then, cross-entropy loss is used, which now comes with significant improvements by taking into account such interactions. This is recently improved [54] by replacing the propagation module with an attention model. The relation between CE loss and some of the widely used pairwise losses is studied from a mutual information point of view [5]. CE loss is viewed as approximate bound-optimization for minimizing pairwise losses; CE maximizes mutual information, and so do these pairwise losses, which are reviewed in the following.

Pairwise losses. The first pairwise loss introduced for this task is the so-called contrastive loss [18], where embeddings of relevant pairs are pushed as close as possible, while those of non-relevant ones are pushed far enough. Since the target task is typically a ranking one, the triplet loss [53], a popular and widely used loss, improves that by forming training triplets in the form of anchor, positive and negative examples. The loss is a function of the difference between anchor-to-positive and anchor-to-negative distances and is zero if such a difference is large enough, therefore satisfying the objectives of a ranking task for this triplet. Optimization over all pairs or triplets is not tractable and is observed to be sub-optimal [69]. As a result, a lot of attention is paid to finding informative pairs and triplets [32, 36, 51, 55, 56], which typically includes heuristics. Several other losses are suggested in the literature [56, 65, 69] and are added to the long list of hand-designed proxy losses which target to learn embeddings that transfer well to a ranking or a similar task.

A few cases follow a principled approach for obtaining a loss that is appropriate for ranking tasks. This is the case with the work of Ustinova *et al.* [60] where the goal is to minimize the probability that the similarity between embeddings of a non-relevant pair is larger than that of a relevant one. This probability is approximated by the quantization of the range of possible similarities and the histogram loss, which is estimated within a single batch. Their work dispenses with the need for any kind of sampling for mini-batch construction. An information-theoretic loss function, called RankMI [25], maximizes the mutual information between the samples within the same semantic class using a neural network. Another principled approach focuses on optimizing AP, which is a standard retrieval evaluation metric. A smooth approximation of it is often used in the literature [19, 47, 49], while the work of Brown *et al.* [6] is the closest to ours. In combination with such AP-based losses, a large batch size is crucial, which meets the limitations set by the hardware. Such limitations are overcome in the work of Revaud *et al.* [47] who uses a batch of $4,000$ high-resolution images.

Embedding mixup. Manifold mixup [63], which involves mixing [72] intermediate representations and labels of two examples, has demonstrated to improve generalizability for supervised learning by encouraging smoother decision boundaries. Such techniques are investigated for embedding learning and image retrieval by mixing the embedding of two examples. Duan *et al.* [11] uses adversarial training to synthesize additional negative samples from the observed negatives. Kalantidis *et al.* [24] synthesize hard-negatives for contrastive self-supervised learning by mixing the embedding of the two hardest negatives and also mixing them with the query itself. Zheng *et al.* [74] uses a linear interpolation between the embeddings to manipulate the hardness levels. In the work of Gu *et al.* [15], two embedding vectors from the same class are used to generate symmetrical synthetic examples and hard-negative mining is performed within the set of original and the synthetic examples. This is further extended to proxy-based losses, where the embedding of examples from different classes and labels is mixed to generate synthetic proxies [16]. Linearly interpolating labels entails the risk of generating false negatives if the interpolation factor is close to 0 or 1. Such limitations are overcome in the work of Venkataramanan *et al.* [62], which generalizes mixing examples from different classes for pairwise loss functions. The proposed *SiMix* approach differs from the aforementioned techniques as it operates on the similarity scores instead of the embedding vectors, does not require training an additional model, making it computationally efficient. Furthermore, unlike the existing mixup techniques, it uses a synthetic sample in the roles of a query, positive and negative example.

3. Method

This section presents the task of image retrieval and the proposed approach for learning image embeddings.

Task. We are given a query example $q \in \mathcal{X}$ and a collection of examples $\Omega \subset \mathcal{X}$, also called database, where $\mathcal{X}$ is the space of all images. The set of database examples that are positive or negative to the query are denoted by P_q and N_q, respectively, with $\Omega = P_q \cup N_q$. Ground-truth information for the positive and negative sets per query is obtained according to discrete class labels per example, *i.e.* if two examples come from the same class, then they are considered positive to each other, otherwise negative. This is the case for all (training or testing) databases used in this work. Terms example and image are used interchangeably in the following text. In image retrieval, all database images are ranked according to similarity to the query q, and the goal is to rank positive examples before negative ones.

Deep image embeddings. Image embeddings, otherwise called descriptors, are generated by function $f_\theta : \mathcal{X} \rightarrow \mathbb{R}^d$. In this work, function f_θ is a deep fully convolutional neural network or a vision transformer mapping input images of any size or aspect ratio to an L_2-normalized d-dimensional embedding. Embedding for image x is denoted by $\mathbf{x} = f_\theta(x)$. Parameter set θ of the network is learned during the training. Similarity between a query q and a database image x is computed by the dot product of the corresponding embeddings and is denoted by $s(q, x) = \mathbf{q}^\top \mathbf{x}$, also denoted as s_{qx} for brevity.

Evaluation metric. Recall@k is one of the standard metrics to evaluate image retrieval methods. For query q, it is defined as a ratio of the number of relevant (positive) examples within the top-k ranked examples to the total number of relevant examples for q given by $|P_q|$. It is denoted by $R^k_\Omega(q)$ when computed for query q and database Ω and can be expressed as

$$R^k_\Omega(q) = \frac{\sum_{x \in P_q} H(k - r_\Omega(q,x))}{|P_q|}, \tag{1}$$

where $r_\Omega(q,x)$ is the rank of example x when all database examples in Ω are ranked according to similarity to query q. Function $H(.)$ is the Heaviside step function, which is equal to 0 for negative values, otherwise equal to 1. The rank of example x is computed by

$$r_\Omega(q,x) = 1 + \sum_{z \in \Omega, z \neq x} H(s_{qz} - s_{qx}), \tag{2}$$

Therefore, (1) can now be expressed as

$$R^k_\Omega(q) = \frac{\sum_{x \in P_q} H(k - 1 - \sum_{z \in \Omega, z \neq x} H(s_{qz} - s_{qx}))}{|P_q|}. \tag{3}$$

Recall@k surrogate loss. The computation of recall in (3) involves the use of the Heaviside step function. The gradient of the Heaviside step function is a Dirac delta function. Hence, direct optimization of recall with back-propagation is not feasible. A common smooth approximation of the Heaviside step function is provided by the logistic function [21, 22, 28], a common sigmoid function $\sigma_\tau : \mathbb{R} \to \mathbb{R}$ controlled by temperature τ, which is given by

$$\sigma_\tau(u) = \frac{1}{1 + e^{-\frac{u}{\tau}}}, \tag{4}$$

where large (small) temperature value leads to worse (better) approximation and denser (sparser) gradient. This approximation is common in the machine learning literature for several tasks [17, 33, 52] and also appears in the approximation of the Average Precision evaluation metric [6], which is used for the same task as ours. By replacing the step function with the sigmoid function, a smooth approximation of recall is obtained as

$$\tilde{R}^k_\Omega(q) = \frac{\sum_{x \in P_q} \sigma_{\tau_1}(k - 1 - \sum_{\substack{z \in \Omega \\ z \neq x}} \sigma_{\tau_2}(s_{qz} - s_{qx}))}{|P_q|}, \tag{5}$$

which is differentiable and can be used for training with back-propagation. The two sigmoids have different function domains and, therefore, different temperatures (see Figure 2). The minimized single-query loss in a mini-batch B, with size $M = |B|$, and query $q \in B$ is given by

$$L^k(q) = 1 - \tilde{R}^k_{B \setminus q}(q). \tag{6}$$

while incorporation of multiple values of k is performed in the loss given by

$$L^K(q) = \frac{1}{|K|} \sum_{k \in K} L^k(q). \tag{7}$$

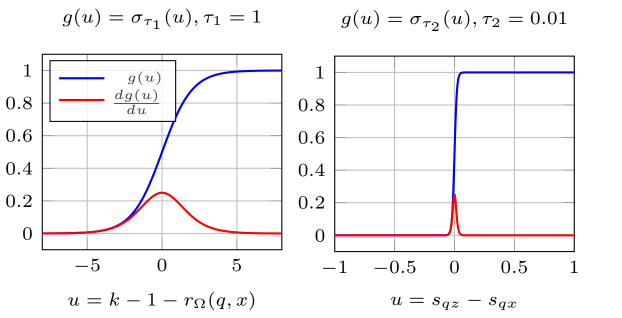

Figure 2. The two sigmoid functions which replace the Heaviside step function for counting the positive examples in the short-list of size k (left) and for estimating the rank of examples (right).

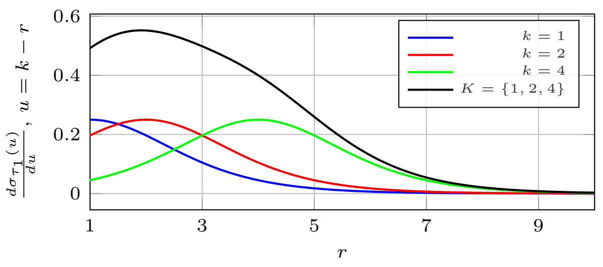

Figure 3. Gradient magnitude of the sigmoid used to count the positive examples in the short-list of size k versus the rank r (equal to $r_\Omega(q,x)$, see (2)) of a positive example x. It shows how much a positive example is pushed towards lower ranks depending on its current rank. In the case of multiple values for k, the total gradient is equivalent to the sum of the separate ones.

Figure 3 shows the impact of using single or multiple values for k.

All examples in the mini-batch are used as queries and the average loss over all queries is minimized during the training. The proposed loss is referred to as *Recall@k Surrogate loss*, or RS@k loss for brevity.

To allow for 0 loss when k is smaller than the number of positives (note that exact recall@k is less than 1 by definition), we slightly modify (5) during the training. Instead of dividing by $|P_q|$, we divide by $\min(k, |P_q|)$, and, consequently, we clip values larger than k in the numerator to avoid negative loss values.

Similarity mixup (SiMix). Given original batch B, virtual batch $\hat{B}$ is created by mixing all pairs of positive examples in the original batch. Embeddings of examples $x \in B$ and $z \in B$ are used to generate mixed embedding

$$\mathbf{v}_{xz\alpha} = \alpha\mathbf{x} + (1 - \alpha)\mathbf{z} \quad | \quad \alpha \sim U(0,1), \tag{8}$$

for a virtual example that is denoted by $xz\alpha \in \hat{B}$. The similarity of an original example $w \in B$ to the virtual example $xz\alpha \in \hat{B}$ is given by

$$s(w, xz\alpha) = \mathbf{w}^\top \mathbf{v}_{xz\alpha} = \alpha s_{wx} + (1 - \alpha) s_{wz}, \tag{9}$$

where the original and virtual examples can be the query and database examples, respectively, or vice versa. In case both examples are virtual, *e.g.* $xz\alpha_1 \in \hat{B}$ used as a query and $yw\alpha_2 \in \hat{B}$ as a part of the database, then their similarity is given by

$$\begin{aligned} s(xz\alpha_1, yw\alpha_2) &= \mathbf{v}_{xz\alpha_1}^\top \mathbf{v}_{yw\alpha 2} \\ &= \alpha_1\alpha_2 s_{xy} + (1-\alpha_1)(1-\alpha_2)s_{zw} \\ &+ \alpha_1(1-\alpha_2)s_{xw} + (1-\alpha_1)\alpha_2 s_{zy}. \end{aligned} \tag{10}$$

The pairwise similarities that appear on the right-hand side of the previous formulas, *e.g.* s_{wx} and s_{wz} in (9), are computed from the embeddings of the original, non-virtual examples and are also required for the computation of the RS@k without any virtual examples. Therefore, the mini-batch is expanded to $B \cup \hat{B}$ by adding virtual examples without the need for explicit construction of the corresponding embeddings or computation of the similarity via dot product; simple mixing of the corresponding pairwise scalar similarities is enough. SiMix reduces to mixing pairwise similarities due to the lack of re-normalization of the mixed embeddings, which is different to existing practice in prior work [15, 16, 24, 62] and brings training efficiency benefits.

Virtual examples are created only between examples of the same classes and are labeled according to the class of the original examples that are mixed. Virtual examples are used both as queries and as database examples, while mixing is applied to all pairs of positive examples inside a mini-batch.

Overview. An overview of the training process with the proposed loss and SiMix is given in Algorithm 1. In case SiMix is not used, then lines 11, 13, 14 and 15 are skipped. It is assumed that each image in training is labeled to a class. Mini-batches of size M are generated by randomly sampling m images per class out of M/m sampled classes.

Algorithm 1 Training with RS@k and SiMix.

```
1:  procedure TRAIN-RS@K($X, Y, M, m$)
2:      $X$ : training images
3:      $Y$ : class labels
4:      $M$ : mini-batch size
5:      $m$ : number of images per class in mini-batch
6:
7:      $\theta \leftarrow$ initialize according to pre-training          ▷ use ImageNet
8:      for iteration $\in [1, \dots,$ number-of-iterations$]$ do
9:          $loss \leftarrow 0$                                            ▷ set batch loss to zero
10:         $B \leftarrow$ BATCH-SAMPLER($X, Y, M, m$)
11:         $\hat{B} \leftarrow$ VIRTUAL-BATCH($B$)                          ▷ enumerate virtual examples
12:         for $(x, z) \in B \times B$ do compute $s(x, z)$                ▷ use $\mathbf{x}^\top\mathbf{z}$
13:         for $(x, z) \in B \times \hat{B}$ do compute $s(x, z)$          ▷ use (9)
14:         for $(x, z) \in \hat{B} \times \hat{B}$ do compute $s(x, z)$    ▷ use (10)
15:         $B \leftarrow B \cup \hat{B}$                                    ▷ expand batch with virtual examples
16:         for $q \in B$ do                                                ▷ use each image in the batch as query
17:             $loss \leftarrow loss + L^K(q)$                              ▷ Recall@k loss (7)
18:         end for
19:         $\theta \leftarrow$ MINIMIZE($\frac{loss}{|B|}$)                 ▷ SGD update
20:     end for
21: end procedure
```

Dataset	#Images	#Classes	#Avg
iNaturalist Train [61]	325,846	5,690	57.3
iNaturalist Test [61]	136,093	2,452	55.5
VehicleID Train [29]	110,178	13,134	8.4
VehicleID Test [29]	40,365	4,800	8.4
SOP Train [39]	59,551	11,318	5.3
SOP Test [39]	60,502	11,316	5.3
Cars196 Train [27]	8,054	98	82.1
Cars196 Test [27]	8,131	98	82.9
$\mathcal{R}$Oxford [45]	4,993	11	n/a
$\mathcal{R}$Paris [45]	6,322	11	n/a
GLDv1 [38]	1,060,709	12,894	82.3

Table 1. Dataset composition for training and evaluation.

4. Experiments

4.1. Datasets

The training and evaluation is performed on four widely used image retrieval benchmarks, namely iNaturalist [61], PKU VehicleID [29], Stanford Online Products [39] (SOP) and Stanford Cars [27] (Cars196). Recall at top k retrieved images, denoted by r@k, is one of the standard evaluation metrics in these benchmarks. Metric r@k is 1 if at least one positive image appears in the top k list, otherwise 0. The metric is averaged across all queries. Note that this is different from the standard definition of recall in (1).

iNaturalist [61] is firstly used by Brown *et al.* [6], whose setup we follow: 5,690 classes for training and 2,452 classes for testing. For VehicleID, according to the standard setup [29], 13,134 classes are used for training, and the evaluation is conducted on the predefined small (800 classes), medium (1600 classes) and large (2400 classes) test sets. For SOP [39] and Cars196 [27], the standard experimental setup of Song *et al.* [56] is followed. The first half of the classes are used for training and the rest for testing, resulting in 11,318 classes for SOP and 98 for Cars196.

The method is evaluated for instance-level search on Revisited Oxford ($\mathcal{R}$Oxford) and Paris ($\mathcal{R}$Paris) benchmark [45], where the evaluation metric is mean Average Precision (mAP). The training uses the Google Landmarks dataset (GLDv1) [38] to perform a comparison with the work of Revaud *et al.* [47] and their AP loss. The validation is performed according to the work of Tolias *et al.* [59].

The number of examples, classes, and average number of examples per class can be found in Table. 1. Note that these datasets are diverse in the number of training examples, the number of classes, and the number of examples per class, ranging from class balanced [27] to long-tailed [61].

4.2. Implementation details

Implementation details are identical for the four image retrieval benchmarks but differ for $\mathcal{R}$Oxford/$\mathcal{R}$Paris to follow and compare to prior work [47]. Differences are clarified when needed.

Architecture. An ImageNet [9] pre-trained ResNet-50 [20] is used as the backbone for deep image embeddings. Building on the standard implementation of [51], the BatchNorm parameters are kept frozen during the training. After the convolutional layers, Generalized mean pooling [46] and layer normalization [1] are used, similar to [58]. For vision transformers [10] ViT-B/32 and ViT-B/16 with an ImageNet-21k initialization from the timm library [68] are used. The last layer of the model is a d dimensional fully connected (FC) layer with L_2 normalization. In the case of $\mathcal{R}$Oxford/$\mathcal{R}$Paris, ResNet-101 [20] is used, layer normalization is not added, while the FC layer is initialized with the result of whitening [46].

Training hyper-parameters. For ResNet architectures, Adam optimizer [26] is used and for vision transformers, AdamW [31] is used. This paper follows the standard class-balanced-sampling [6, 36, 58] with 4 samples per class for all the datasets, while classes with less than 4 samples are not used for training. Unless stated otherwise, the batch size for training is set $4,000$ for all datasets but Cars196 where it is equal to $4 \times \#\text{classes} = 392$. Following the setup of ProxyNCA++ [58], the training set is split into training and validation by using the first half of the classes for training and the other half for validation. With this split, a grid search determines the learning rate, decay steps, decay size and the total number of epochs. Once the hyper-parameters are fixed, training is conducted once on the entire training set and evaluated on the test set. When training on GLDv1 and testing on $\mathcal{R}$Oxford/$\mathcal{R}$Paris, the batch size is set to 4096 [47], and training is performed for 500 batches, while other training hyper-parameters are set as in the work and GitHub implementation of Radenovic *et al*. [46]. Note that the hyper-parameters for each dataset will be released with the implementation.

RS@k hyper-parameters. The proposed Recall@k Surrogate (RS@k) loss (5) contains three hyper-parameters: sigmoid temperature τ_2 - applied on similarity differences, sigmoid temperature τ_1 - applied on ranks and the set of values for k for which the loss is computed. Both sigmoid temperatures are kept fixed across all the experiments as $\tau_2 = 0.01$ (same as [6]) and $\tau_1 = 1$. The values of k are kept fixed as $k = \{1, 2, 4, 8, 16\}$ without SiMix and $k = \{1, 2, 4, 8, 12, 16, 20, 24, 28, 32\}$ with SiMix. For GLDv1 [38], this is $k = \{1, 2, 4\}$, and $k = \{1, 2, 4, 8\}$, respectively. The values of k are studied in the supplementary materials and the sigmoid temperature τ_1 are investigated in Section 4.4, where it is observed that the method is not very sensitive to these hyper-parameters.

Large batch size. To dispense with the GPU hardware constraints and manage to train with the large batch size, we follow the multistage back-propagation of Revaud *et al*. [47]. A forward pass is performed to obtain all embeddings while intermediate tensors are discarded from memory. Then, the loss is computed, and so are the gradients w.r.t. the embeddings. Finally, each of the embeddings is recomputed, this time allowing the propagation of the gradients. Note that there is no implementation online of this approach and that the code of this work will become publicly available. Algorithm 1 does not include such implementation details, but it is compatible with such an extension. The batch-size impact for the proposed RS@k loss function is validated in Section 4.4.

Discussion. The methods in the literature use different embedding sizes, d, therefore, the models for the RS@k loss are trained with two embedding sizes of $d = 128$ and $d = 512$ for image retrieval benchmarks [27,29,39,61], and $d = 2048$ for $\mathcal{R}$Oxford/$\mathcal{R}$Paris [45], to allow a fair comparison. In the standard split, the image retrieval benchmarks [27, 29, 39, 61] do not contain an explicit validation set; as a result, image retrieval methods often tune the hyper-parameters on the test set, leading to the issue of training with test set feedback. This issue has been studied in [36], which proposes to train different methods with identical hyper-parameters. The setup of [36] is not directly usable for experiments with the RS@k loss, as large batch sizes are crucial to estimate recall@k accurately. Furthermore, their setup does not allow mixup. Therefore, instead of following [36], the issue is eliminated by using a part of the training set for validation as described above.

4.3. Evaluation

Unless otherwise stated, the results of the competing methods are taken from the original papers. Methods marked with a † were trained by the authors of this paper, using the same implementation as used for the RS@k loss. The results on image retrieval benchmarks [27, 29, 39, 61] are compared with the methods that use either ResNet-50 [20] or Inception network [57]. ResNet-50 [20] is represented as R^d_{50} in the tables and the standard Inception network [57] as I^d_1, the Inception network with BatchNorm as I^d_3 (same as [58]). Here d is the embedding size. On all the datasets, the performance of the baseline, Smooth-AP (SAP) [6], is also reported with Generalized mean pooling [46] and layer normalization [1], shown as SAP† (+Gem +LN). This is to eliminate any performance boost in the comparisons that were caused by the architecture. Note that unless otherwise stated in our experiments, the batch size for SAP is set as 384, the same as the original implementation [6]. Further, we demonstrate the performance of SAP and RS@k on ViT-B architectures. The variant of ViT-B that uses a patch size of 32×32 is denoted by ViT-B/32 and

Method	Arch.^{dim}	iNaturalist [61]				SOP [39]				VehicleID [29]						Cars196 [27]			
		r@k																	
										Small		Medium		Large					
		1	4	16	32	10^0	10^1	10^2	10^3	1	5	1	5	1	5	1	2	4	8
ProxyNCA [35]	I_1^{128}	61.6	77.4	87.0	90.6	73.7	-	-	-	-	-	-	-	-	-	73.2	82.4	86.4	88.7
Margin [69]	R_{50}^{128}	58.1	75.5	86.8	90.7	72.7	86.2	93.8	98.0	-	-	-	-	-	-	79.6	86.5	91.9	95.1
Divide [53]	R_{50}^{128}	-	-	-	-	75.9	88.4	94.9	98.1	87.7	92.9	85.7	90.4	82.9	90.2	-	-	-	-
MIC [50]	R_{50}^{128}	-	-	-	-	77.2	89.4	95.6	-	86.9	93.4	-	-	82.0	91.0	-	-	-	-
Cont. w/M [67]	R_{50}^{128}	-	-	-	-	80.6	91.6	96.2	98.7	94.7	96.8	93.7	95.8	93.0	95.8	-	-	-	-
RS@k†	R_{50}^{128}	69.3	82.9	90.6	93.1	80.6	91.6	96.4	**98.8**	**95.6**	**97.8**	**94.4**	**96.8**	**93.5**	**96.6**	78.1	85.8	91.1	94.5
RS@k† +SiMix	R_{50}^{128}	**69.6**	**83.3**	**91.2**	**93.8**	**80.9**	**91.7**	**96.5**	**98.8**	95.4	97.5	93.8	96.6	93.0	96.2	**84.7**	**90.9**	**94.7**	**96.9**
		+21%	+26%	+32%	+33%	+1.5%	+1.2%	+7.9%	+7.7%	+17%	+31%	+11%	+24%	+7.1%	+19%	+25%	+33%	+35%	+37%
FastAP [7]	R_{50}^{512}	60.6	77.0	87.2	90.6	76.4	89.0	95.1	98.2	91.9	96.8	90.6	95.9	87.5	95.1	-	-	-	-
MS [66]	I_3^{512}	-	-	-	-	78.2	90.5	96.0	98.7	-	-	-	-	-	-	84.1	90.4	94.0	96.1
NormSoftMax [71]	R_{50}^{512}	-	-	-	-	78.2	90.6	96.2	-	-	-	-	-	-	-	84.2	90.4	94.4	96.9
Blackbox AP [49]	R_{50}^{512}	62.9	79.0	88.9	92.1	78.6	90.5	96.0	98.7	-	-	-	-	-	-	-	-	-	-
Cont. w/M [67]	I_3^{512}	-	-	-	-	79.5	90.8	96.1	98.7	94.6	96.9	93.4	96.0	93.0	96.1	-	-	-	-
HORDE [23]	R_{50}^{512}	-	-	-	-	80.1	91.3	96.2	-	-	-	-	-	-	-	86.2	91.9	95.1	97.2
ProxyNCA++ [58]	R_{50}^{512}	-	-	-	-	80.7	92.5	96.7	98.9	-	-	-	-	-	-	86.5	92.5	95.7	**97.7**
SAP [6]	R_{50}^{512}	67.2	81.8	90.3	93.1	80.1	91.5	96.6	99.0	94.9	97.6	93.3	96.4	91.9	96.2	76.1	84.3	89.8	93.8
SAP† [6] +GeM +LN	R_{50}^{512}	68.7	82.7	90.9	93.5	80.3	92.0	96.9	99.0	94.2	97.2	92.7	96.2	91.0	95.8	78.2	85.6	90.8	94.3
RS@k†	R_{50}^{512}	71.2	84.0	91.3	93.6	**82.8**	**92.9**	**97.0**	99.0	**95.7**	**97.9**	**94.6**	**96.9**	**93.8**	**96.6**	80.7	88.3	92.8	95.7
RS@k† +SiMix	R_{50}^{512}	**71.8**	**84.7**	**91.9**	**94.3**	82.1	92.8	**97.0**	**99.1**	95.3	97.7	94.2	96.5	93.3	96.4	**88.2**	**93.0**	**95.9**	97.4
		+14%	+16%	+16%	+17%	+11%	+5.3%	+12%	+10%	+16%	+13%	+18%	+14%	+11%	+10%	+13%	+6.7%	+4.7%	−13%
SAP† [6]	ViT-B/32^{512}	72.2	84.6	91.6	93.9	83.7	94.0	97.8	99.3	94.8	97.7	93.5	96.8	92.1	96.3	78.1	85.7	91.0	94.8
RS@k†	ViT-B/32^{512}	75.9	87.1	93.1	95.1	85.1	94.6	98.0	99.3	95.1	97.7	94.1	96.7	93.2	96.5	78.1	86.4	92.3	95.6
SAP† [6]	ViT-B/16^{512}	79.1	89.0	94.2	95.8	86.6	95.4	98.4	99.5	95.5	97.7	94.2	96.9	93.1	96.6	86.2	92.1	95.1	97.2
RS@k†	ViT-B/16^{512}	83.9	92.1	95.9	97.2	88.0	96.1	98.6	99.6	96.2	98.0	95.2	97.2	94.7	97.1	89.5	94.2	96.6	98.3

Table 2. Recall@k(%) on iNaturalist [61], Stanford Online Products (SOP) [39], PKU VehicleID [29] and Stanford Cars (Cars196) [27]. Best results are shown with **bold**, previous state-of-the-art with underline and relative gains over the state-of-the-art in % of error reduction with blue and relative declines in red. Methods marked with † were trained using the same pipeline by the authors of this paper.

the one that uses a patch size of 16×16 by ViT-B/16.

iNaturalist. The results on iNaturalist [61] species recognition are presented in Table 2. The performances of the competing methods are taken from [6], which uses the official implementations of these methods. It can be clearly seen that the RS@k outperforms classification and pairwise losses, including the three AP approximation losses, reaching the recall@1 score of 71.8% with SiMix, an error reduction of 14%.

SOP. The performance on SOP [39] is presented in Table 2, along with the comparisons with the competing methods. The proposed RS@k loss demonstrates clear state-of-the-art results, surpassing ProxyNCA++ [58] by 2.0% on recall@1, an error reduction of 10.4%. If a smaller batch size, equal to 384, is used for RS@k, it reaches a performance of 81.2%, 92.2%, 96.9% and 99.0% on r@10^0, r@10^1, r@10^2 and r@10^3 respectively. This result shows that large batch size helps in improving the performance, but RS@k outperforms the competing methods even with smaller batch size.

VehicleID. The results on VehicleID [29] are presented in Table 2. RS@k outperforms the competing methods both with and without SiMix. Better results were observed without SiMix where RS@k reaches recall@1 performance of 95.7%, 94.6% and 93.8% on the small, medium, and large test sets, respectively.

Cars196. Evaluation on a small scale dataset, Cars196 [27] is presented in the Table 2. We train SAP with a batch size of 392; it provides a performance of 79.5%, 86.6%, 91.2%, and 94.4% and when combined with SiMix a performace of 85.4%, 91.0%, 94.3% and 96.7% on r@1, r@2, r@4 and r@8 respectively. SiMix makes a large difference in performance for both RS@k and SAP [6], primarily because of a smaller batch size (392), as constrained by the low number of classes. With SiMix, RS@k reaches the state-of-the-art results on three out of four recall@k values. If the batch size is further increased to 588 by changing the number of samples per class from 4 to 6, then RS@k provides a larger gain with performance 88.3%, 93.3%, 95.9% and 97.6%.

Results with ViT-B. The results by replacing the ResNet-50 [20] backbone with a ViT-B [10] for SAP [6] and the proposed RS@k are also shown in Table 2. With an exception of ViT-B/32 on VehicleID and Cars196 datasets, the use of ViT-B backbone leads to better performance for both methods, compared to the ResNet counterpart. It can be clearly seen that RS@k outperforms SAP [6] on all datasets. ViT-B/16 when trained with RS@k shows unprecedented performance on all datasets reaching recall@1 score of 83.9% on iNaturalist [61], 88.0% on SOP [39], 96.2% on VehicleID [29] (small) and 89.5% on Cars196 [27]. Note that while ResNet-50 has 24.5 M parameters and operates with 8.12 GMac/image, ViT-B has 87.8 M parameters and operates with 4.36 and 16.8 GMac/image for ViT-B/32 and ViT-B/16 respectively.

Concurrent work. The method of learning intra-batch

Arch.	Loss	Train-set		Mean		$\mathcal{R}$O		$\mathcal{R}$O+$\mathcal{R}$1M		$\mathcal{R}$Par		$\mathcal{R}$P+$\mathcal{R}$1M	
				all	$\mathcal{R}$1M	med	hard	med	hard	med	hard	med	hard
GeM$*$	AP [19]	Landmarks-clean [2] [14]	[47]/ [59]	49.7	36.7	67.1	42.3	47.8	22.5	80.3	60.9	51.9	24.6
GeM$*$	AP [19]	GLDv1 [38]	[47]/github	-	-	66.3	42.5	-	-	80.2	60.8	-	-
GeM†	SAP [6]	GLDv1 [38]	[6]	52.7	40.6	67.9	46.3	49.5	25.8	81.7	63.3	57.4	29.8
GeM†	RS@k	GLDv1 [38]	ours	53.1	41.0	68.3	46.1	50.1	25.8	82.1	63.9	57.9	30.2
GeM+SiMix†	RS@k	GLDv1 [38]	ours	53.1	41.8	68.4	45.3	51.0	26.4	81.2	62.4	58.7	31.1

Table 3. Performance comparison (mAP%) on $\mathcal{R}$Oxford and $\mathcal{R}$Paris with 1m distractor images ($\mathcal{R}$1m). Mean performance is reported across all setups or the large-scale setups only. $*$ denotes that the FC layer is not part of the training but is added afterward to implement whitening. Batch size is 4096 for all methods; SiMix virtually increases it to 10240. ResNet101 is used as a backbone for all methods.

connections for deep metric learning [54] achieves r@1 of 81.4% on the SOP and 88.1% on Cars196 dataset. The approach for Grouplet embedding learning [73] obtains r@1 of 82.0% on SOP and 91.5% on Cars196. The metric mixup approach [62] reports the best results of 81.3% r@1 on SOP in combination with ProxyNCA++ [58] and 89.6% on Cars196 which is in combination with MS [66].

$\mathcal{R}$Oxford/$\mathcal{R}$Paris. Table 3 summarizes a comparison with AP-based losses in the literature on $\mathcal{R}$Oxford/$\mathcal{R}$Paris with and without distractor images. The comparison is performed with GLDv1 as a training set whose performance is reported for the work of Revaud *et al.* [47] in their GitHub page, while the *landmarks-clean dataset* is avoided as all initial images are not publicly available at the moment. During the training performed by us, training images are down-sampled to have a maximum resolution of 1024×1024. The inference is performed with multi-resolution descriptors at three scales with up-sampling and down-sampling by a factor of $\sqrt{2}$. Note that SAP is not evaluated on these datasets in the original work and this experiment is performed by us, which outperforms the previously used AP loss [19]. RS@k, with or without the SiMix, increases the performance by a small margin.

4.4. Effect of hyper-parameters

We study the impact of hyper-parameter on the Cars196 dataset [27] since it is the smallest compared to the others and has the lowest training time.

Sigmoid temperature τ_1 - applied on ranks. The effect of the sigmoid temperature τ_1 is summarized in Figure 4 (left). For both setups of with and without SiMix, $\tau_1 = 1.0$ gives best results while higher and lower values lead to a decline.

Batch size. The effect of the varying batch size is shown in Figure 4 (right). It demonstrates that large batch size leads to better results. A significant performance boost is observed with the use of SiMix, especially in the small batch size regime, which comes at a small extra computation. A comparison with SAP [6] is also shown in this figure. Note that on smaller batch sizes, the proposed RS@k outperforms SAP with a larger margins.

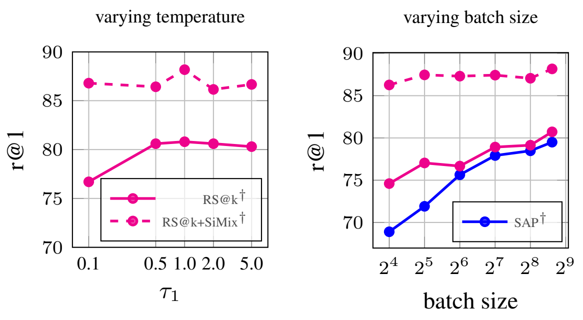

Figure 4. The effect of sigmoid temperature τ_1 applied on ranks (left) and of batch size (right). Results are shown on Cars196 [27].

5. Conclusions

This work has presented image embedding learning for retrieval using a novel surrogate loss function for the recall@k metric. State-of-the-art results were achieved on a number of standard benchmarks. Training with very large batch size, up to 4k images, has shown to be highly beneficial. The batch size is further increased, in a virtual way, with a newly proposed mixup approach that acts directly on the scalar similarities. This approach offers a boost in performance at a small increase of the computational cost, while its applicability goes beyond the proposed loss. The implementation of the proposed Recall@k Surrogate loss, proposed similarity mixup, along with the training procedure that allows the use of large batch sizes on a single GPU by sidestepping memory constraints, is available at https://github.com/yash0307/RecallatK_surrogate.

Acknowledgements

The authors thank Kristína Cinová for proofreading. This research was supported by Research Center for Informatics (project CZ.02.1.01/0.0/0.0/16_019/0000765 funded by OP VVV), by the Grant Agency of the Czech Technical University in Prague, grant No. SGS20/171/OHK3/3T/13, by Project StratDL in the realm of COMET K1 center Software Competence Center Hagenberg, Amazon Research Award, and Junior Star GACR, grant No. GM 21-28830M.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 6

[2] Artem Babenko, Anton Slesarev, Alexandr Chigorin, and Victor Lempitsky. Neural codes for image retrieval. In *ECCV*, 2014. 8

[3] Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. An actor-critic algorithm for sequence prediction. In *ICLR*, 2017. 1, 2

[4] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. In *ICLR*, 2018. 1

[5] Malik Boudiaf, Jérôme Rony, Imtiaz Masud Ziko, Eric Granger, Marco Pedersoli, Pablo Piantanida, and Ismail Ben Ayed. Metric learning: cross-entropy vs. pairwise losses. In *ECCV*, 2020. 2

[6] Andrew Brown, Weidi Xie, Vicky Kalogeiton, and Andrew Zisserman. Smooth-ap: Smoothing the path towards large-scale image retrieval. In *ECCV*, 2020. 1, 2, 3, 4, 5, 6, 7, 8

[7] Fatih Cakir, Kun He, Xide Xia, Brian Kulis, and Stan Sclaroff. Deep metric learning to rank. In *CVPR*, 2019. 2, 7

[8] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *CVPR*, 2019. 2

[9] Wei Dong, Richard Socher, Li Li-Jia, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009. 6

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 2, 6, 7

[11] Yueqi Duan, Wenzhao Zheng, Xudong Lin, Jiwen Lu, and Jie Zhou. Deep adversarial metric learning. In *CVPR*, 2018. 3

[12] Ismail Elezi, Sebastiano Vascon, Alessandro Torcinovich, Marcello Pelillo, and Laura Leal-Taixé. The group loss for deep metric learning. In *ECCV*, 2020. 2

[13] Martin Engilberge, Louis Chevallier, Patrick Pérez, and Matthieu Cord. Sodeep: a sorting deep net to learn ranking loss surrogates. In *CVPR*, 2019. 2

[14] Albert Gordo, Jon Almazan, Jerome Revaud, and Diane Larlus. End-to-end learning of deep visual representations for image retrieval. *IJCV*, 2017. 8

[15] Geonmo Gu and Byungsoo Ko. Symmetrical synthesis for deep metric learning. In *AAAI*, 2020. 3, 5

[16] Geonmo Gu, Byungsoo Ko, and Han-Gyu Kim. Proxy synthesis: Learning with synthetic classes for deep metric learning. *AAAI*, 2021. 3, 5

[17] Shixiang Gu, Sergey Levine, Ilya Sutskever, and Andriy Mnih. Muprop: Unbiased backpropagation for stochastic neural networks. In *ICLR*, 2016. 4

[18] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR*, 2006. 2, 3

[19] Kun He, Yan Lu, and Stan Sclaroff. Local descriptors optimized for average precision. In *CVPR*, 2018. 2, 3, 8

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 6, 7

[21] A Iliev, Nikolay Kyurkchiev, and S Markov. On the approximation of the step function by some sigmoid functions. *Mathematics and Computers in Simulation*, 2017. 4

[22] Anton Iliev Iliev, Nikolay Kyurkchiev, and Svetoslav Markov. On the approximation of the cut and step functions by logistic and gompertz functions. *Biomath*, 2015. 4

[23] Pierre Jacob, David Picard, Aymeric Histace, and Edouard Klein. Metric learning with horde: High-order regularizer for deep embeddings. In *ICCV*, 2019. 7

[24] Yannis Kalantidis, Mert Bulent Sariyildiz, Noe Pion, Philippe Weinzaepfel, and Diane Larlus. Hard negative mixing for contrastive learning. *NeurIPS*, 2020. 3, 5

[25] Mete Kemertas, Leila Pishdad, Konstantinos G Derpanis, and Afsaneh Fazly. Rankmi: A mutual information maximizing ranking loss. In *CVPR*, 2020. 3

[26] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 6

[27] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *ICCV workshops*, 2013. 2, 5, 6, 7, 8

[28] Nikolay Kyurkchiev and Svetoslav Markov. Sigmoid functions: some approximation and modelling aspects. *LAP LAMBERT Academic Publishing, Saarbrucken*, 2015. 4

[29] Hongye Liu, Yonghong Tian, Yaowei Wang, Lu Pang, and Tiejun Huang. Deep relative distance learning: Tell the difference between similar vehicles. In *CVPR*, 2016. 5, 6, 7

[30] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In *CVPR*, 2017. 2

[31] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 6

[32] Jing Lu, Chaofan Xu, Wei Zhang, Ling-Yu Duan, and Tao Mei. Sampling wisely: Deep image embedding by top-k precision optimization. In *ICCV*, 2019. 3

[33] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *ICLR*, 2017. 4

[34] Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Conditional probability models for deep image compression. In *CVPR*, 2018. 1

[35] Yair Movshovitz-Attias, Alexander Toshev, Thomas K Leung, Sergey Ioffe, and Saurabh Singh. No fuss distance metric learning using proxies. In *ICCV*, 2017. 2, 7

[36] Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. A metric learning reality check. In *ECCV*, 2020. 2, 3, 6

[37] Gattigorla Nagendar, Digvijay Singh, Vineeth N Balasubramanian, and CV Jawahar. Neuro-iou: Learning a surrogate loss for semantic segmentation. In *BMVC*, 2018. 1

[38] Hyeonwoo Noh, Andre Araujo, Jack Sim, Tobias Weyand, and Bohyung Han. Large-scale image retrieval with attentive deep local features. In *ICCV*, 2017. 5, 6, 8

[39] Eng-Jon Ong, Sameed Husain, and Miroslaw Bober. Siamese network of deep fisher-vector descriptors for image retrieval. In *arXiv*, 2017. 2, 5, 6, 7

[40] Yash Patel, Srikar Appalaraju, and R Manmatha. Saliency driven perceptual image compression. In *WACV*, 2021. 1

[41] Yash Patel, Lluis Gomez, Marçal Rusiñol, Dimosthenis Karatzas, and CV Jawahar. Self-supervised visual representations for cross-modal retrieval. In *ICMR*, 2019. 1

[42] Yash Patel, Tomáš Hodaň, and Jiří Matas. Learning surrogates via deep embedding. In *ECCV*, 2020. 1, 2

[43] Yash Patel and Jiri Matas. Feds–filtered edit distance surrogate. In *ICDAR*, 2021. 1

[44] Qi Qian, Lei Shang, Baigui Sun, Juhua Hu, Hao Li, and Rong Jin. Softtriple loss: Deep metric learning without triplet sampling. In *ICCV*, 2019. 2

[45] Filip Radenović, Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondřej Chum. Revisiting oxford and paris: Large-scale image retrieval benchmarking. In *CVPR*, 2018. 2, 5, 6

[46] Filip Radenović, Giorgos Tolias, and Ondřej Chum. Fine-tuning cnn image retrieval with no human annotation. *PAMI*, 2019. 6

[47] Jerome Revaud, Jon Almazán, Rafael S Rezende, and Cesar Roberto de Souza. Learning with average precision: Training image retrieval with a listwise loss. In *ICCV*, 2019. 2, 3, 5, 6, 8

[48] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *CVPR*, 2019. 1

[49] Michal Rolínek, Vít Musil, Anselm Paulus, Marin Vlastelica, Claudio Michaelis, and Georg Martius. Optimizing rank-based metrics with blackbox differentiation. In *CVPR*, 2020. 2, 3, 7

[50] Karsten Roth, Biagio Brattoli, and Bjorn Ommer. Mic: Mining interclass characteristics for improved metric learning. In *ICCV*, 2019. 7

[51] Karsten Roth, Timo Milbich, Samarth Sinha, Prateek Gupta, Bjorn Ommer, and Joseph Paul Cohen. Revisiting training strategies and generalization performance in deep metric learning. In *ICML*, 2020. 3, 6

[52] Ruslan Salakhutdinov and Geoffrey Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 2009. 4

[53] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 2015. 2, 3, 7

[54] Jenny Seidenschwarz, Ismail Elezi, and Laura Leal-Taixé. Learning intra-batch connections for deep metric learning. In *ICML*, 2021. 2, 8

[55] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *NeurIPS*, 2016. 3

[56] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *arXiv*, 2015. 3, 5

[57] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 6

[58] Eu Wern Teh, Terrance DeVries, and Graham W Taylor. Proxynca++: Revisiting and revitalizing proxy neighborhood component analysis. In *ECCV*, 2020. 2, 6, 7, 8

[59] Giorgos Tolias, Tomas Jenicek, and Ondřej Chum. Learning and aggregating deep local descriptors for instance-level recognition. In *ECCV*, 2020. 5, 8

[60] Evgeniya Ustinova and Victor Lempitsky. Learning deep embeddings with histogram loss. In *NeurIPS*, 2016. 3

[61] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *CVPR*, 2018. 2, 5, 6, 7

[62] Shashanka Venkataramanan, Bill Psomas, Yannis Avrithis, Ewa Kijak, Laurent Amsaleg, and Konstantinos Karantzalos. It takes two to tango: Mixup for deep metric learning. In *ICLR*, 2022. 3, 5, 8

[63] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *ICML*, 2019. 3

[64] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. In *CVPR*, 2018. 2

[65] Jian Wang, Feng Zhou, Shilei Wen, Xiao Liu, and Yuanqing Lin. Deep metric learning with angular loss. In *ICCV*, 2017. 3

[66] Xun Wang, Xintong Han, Weilin Huang, Dengke Dong, and Matthew R Scott. Multi-similarity loss with general pair weighting for deep metric learning. In *CVPR*, 2019. 7, 8

[67] Xun Wang, Haozhi Zhang, Weilin Huang, and Matthew R Scott. Cross-batch memory for embedding learning. In *CVPR*, 2020. 7

[68] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019. 6

[69] Chao-Yuan Wu, R Manmatha, Alexander J Smola, and Philipp Krahenbuhl. Sampling matters in deep embedding learning. In *ICCV*, 2017. 2, 3, 7

[70] Jiahui Yu, Yuning Jiang, Zhangyang Wang, Zhimin Cao, and Thomas Huang. Unitbox: An advanced object detection network. In *ACM-MM*, 2016. 1

[71] Andrew Zhai and Hao-Yu Wu. Classification is a strong baseline for deep metric learning. *arXiv preprint arXiv:1811.12649*, 2018. 2, 7

[72] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 2, 3

[73] Yanfu Zhang, Lei Luo, Wenhan Xian, and Heng Huang. Learning better visual data similarities via new grouplet non-euclidean embedding. In *ICCV*, 2021. 8

[74] Wenzhao Zheng, Zhaodong Chen, Jiwen Lu, and Jie Zhou. Hardness-aware deep metric learning. In *CVPR*, 2019. 3