Supervised Metric Learning to Rank for Retrieval via Contextual Similarity Optimization

Christopher Liao [1] **Theodoros Tsiligkaridis** [2] **Brian Kulis** [1]

Abstract

There is extensive interest in metric learning methods for image retrieval. Many metric learning loss functions focus on learning a correct ranking of training samples, but strongly overfit semantically inconsistent labels and require a large amount of data. To address these shortcomings, we propose a new metric learning method, called *contextual loss*, which optimizes *contextual similarity* in addition to cosine similarity. Our contextual loss implicitly enforces *semantic consistency* among neighbors while converging to the correct ranking. We empirically show that the proposed loss is more robust to label noise, and is less prone to overfitting even when a large portion of train data is withheld. Extensive experiments demonstrate that our method achieves a new state-of-the-art across four image retrieval benchmarks and multiple different evaluation settings. Code is available at: https://github.com/Chris210634/metric-learning-using-contextual-similarity

1. Introduction

Image retrieval refers to learning a ranking of instances from a gallery set relative to a query image such that the highest ranked instances are the most relevant to the query. Several real-world applications are powered by this technology, such as person re-identification (Ye et al., 2021), face recognition (Guillaumin et al., 2009), vehicle re-identification (Chu et al., 2019), landmark retrieval (Weyand et al., 2020), and product retrieval (Cakir et al., 2019). Current metric learning techniques often use a dataset with single discrete labels for supervision, and train an embedding space where images with the same label are closer together than images with different labels. However, binary supervision is unreliable, since it does not capture the complexity of relationships in the data. Furthermore, methods which overly rely on the binary supervision can be brittle in the presence of noise, since the supervision is either correct or incorrect. Multi-label datasets (Ranjan et al., 2015) mitigate this problem, but can be expensive to procure, so developing a metric learning method that is *robust to label noise* and *generalizable to test data* is a challenging yet important problem.

Figure 1. Examples of metric learning labels which are inconsistent with semantic information from two standard benchmarks: CUB (top) and SOP (bottom). These labels are caused by a visual feature which is not present or barely visible.

Existing image retrieval approaches fall into two main categories: classification and pairwise ranking losses. *Classification losses* optimize a classifier on top of the embedding layer and discard the classifier at the end of training. *Pairwise ranking losses* train the embedding layer directly by pulling together pairs of samples with the same label and pushing apart pairs of samples with different labels. Pairwise ranking methods include losses which explicitly optimize a ranking metric such as AP (average precision) surrogates: Fast-AP (Cakir et al., 2019), Smooth-AP (Brown et al., 2020), Blackbox AP (Rolínek et al., 2020) and Roadmap (Ramzi et al., 2021). They also include the standard contrastive, triplet and multi-similarity (MS) losses (Wang et al., 2019).

Empirically, classification methods, such as proxy anchor (Kim et al., 2020), proxy NCA (Teh et al., 2020), and HIST (Lim et al., 2022) perform well on small benchmark datasets,

[1]Department of Electrical and Computer Engineering, Boston University [2]MIT Lincoln Laboratory. Correspondence to: Christopher Liao <cliao25@bu.edu>, Theodoros Tsiligkaridis <ttsili@ll.mit.edu>, Brian Kulis <bkulis@bu.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

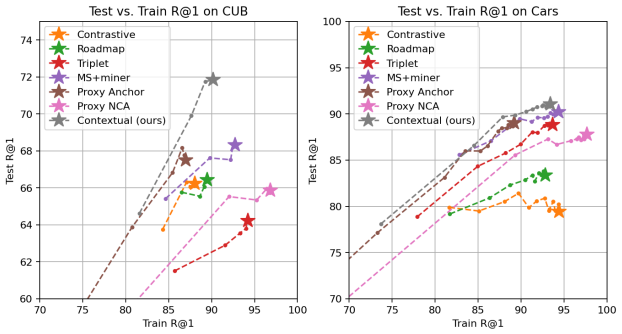

Figure 2. Comparison of our contextual loss with popular metric learning losses. We plot the test R@1 accuracy against the train R@1 accuracy over the course of training on the CUB and Cars benchmarks. The dashed line tracks the R@1 values over the course of training, and the star indicates the R@1 values at the end of training. Compared to baselines, the contextual loss achieves higher test R@1 at the expense of lower train R@1.

while multi-similarity and AP surrogates perform well on large datasets. This general trend is supported by our main results in Section 5. We hypothesize that pairwise ranking methods tend to overfit the training labels while sacrificing semantic consistency of the embedding space. This can be possible even if the labels are "correct", as Figure 1 illustrates. For instance, pulling apart samples of white-necked ravens from common ravens would likely lead to overfitting, since the distinguishing visual attribute is absent from the images. To address this issue, we propose to optimize *contextual similarity* in addition to cosine similarity. The resulting loss function implicitly regularizes the embedding space for semantic consistency among neighbors (see results in Section 4). Figure 2 clearly shows that our method reduces overfitting, since we achieve the best test R@1 accuracy despite lower train R@1 accuracy than some baselines. Results in Section 5 show that our method outperforms all baselines across all standard benchmarks in terms of R@1 accuracy.

Contextual similarity is a widely used evaluation-time technique to boost retrieval accuracy. In simple terms, the contextual similarity is the fraction of neighbors two samples have in common in embedding space. Intuitively, two samples are more likely to share the same label if they have many neighbors in common, regardless of their cosine similarity. Many retrieval frameworks (Zhong et al. (2017), Cao et al. (2020)) use a combination of cosine similarity and contextual similarity for evaluation, but only explicitly optimize the cosine similarity when training. In this paper, we propose to *explicitly* optimize both similarities, since contextual similarity captures crucial semantic information. In another line of work, some unsupervised metric learning methods such as STML (Kim et al., 2022) use contextual similarity to estimate the true similarity between unlabeled samples. Inspired by STML, we show that optimizing contextual similarity directly in the supervised setting is beneficial. As far as we know, we are the first to treat contextual similarity as a loss function for supervised learning. This is non-trivial since contextual similarity involves non-differentiable counting operations, and as a consequence, is not amenable to off-the-shelf optimization techniques. We propose a simple but effective optimization strategy in Section 3, using heuristic gradients. We analytically justify this optimization approach in Section 4.1.

Our contributions are as follows:

1. We introduce the contextual loss, which establishes a new state-of-the-art across all standard image retrieval benchmarks, even when compared to more complicated (e.g. Metrix, HIST and AVSL) and less scalable methods (AP surrogates).

2. Our contextual loss mitigates overfitting by implicitly enforcing semantic consistency among neighbors in the embedding space. As a result, we achieve a 4% improvement in R@1 accuracy over baselines in the presence of label noise.

3. We conduct an extensive experimental study of our method and several popular baselines. This includes empirical results across five different benchmarks, two different experimental settings, accompanied by a comprehensive ablation study. In addition, we tune baselines extensively to promote fair comparisons.

4. The optimization of non-differentiable steps in the loss calculation may be of interest to some readers.

This paper is organized as follows. Section 2 summarizes related work. Section 3 states our method, including how we optimize the non-differentiable steps in calculating contextual similarity. Section 4.1 checks that minimizing the contextual loss corresponds to learning the correct ranking of samples and that the proposed optimization procedure converges. The rest of Section 4 explores why the proposed contextual loss is *less prone to overfitting* than other pairwise ranking losses by analyzing gradients and running targeted experiments. Section 5 and the Appendix present an extensive experimental study. Code is available at: https://github.com/Chris210634/metric-learning-using-contextual-similarity

2. Related Work

Classification Methods We refer to any method which optimizes class centroids in conjunction with embeddings as a classification method. These methods scale with the number of classes in the training set and are usually sensitive to the

learning rate of class centroids (Teh et al., 2020). Classification losses have traditionally performed well on small metric learning benchmarks; these include normalized-softmax (Zhai & Wu, 2018), arcface (Deng et al., 2019), proxy NCA and proxy anchor. More recently, IBC (Seidenschwarz et al., 2021) and HIST (Lim et al., 2022) report an improvement in R@1 when learning a graph neural network in conjunction with class centroids. However, even with these additional tricks, classification methods lag behind pairwise methods on larger benchmarks.

Pairwise Ranking Methods Pairwise ranking losses include the contrastive loss (Hadsell et al., 2006), triplet loss (Weinberger et al., 2005) (Wu et al., 2017), multi-similarity, and AP surrogates (cited in previous section). Despite being more than a decade old, contrastive and triplet losses remain the go-to method for metric learning, and Musgrave et al. (2020a) show that they are comparable in performance to many recent methods. Multi-similarity includes a hard pair mining scheme that is effectively learning to rank. AP maximization methods explicitly learn to rank samples within a mini-batch. AP maximization is challenging because it involves back-propagating through the non-differentiable heaviside function, similar to the current work. As a workaround, Fast-AP uses soft-binning; Smooth-AP uses a low-temperature sigmoid; Roadmap uses an upper bound on the heaviside instead of an approximation. We find that using heuristic gradients works better for optimizing contextual similarity.

Unsupervised Metric Learning The concept of contextual similarity is extensively studied in the unsupervised metric learning literature, mainly in the context of person re-ID (see survey (Ye et al., 2021)). Most unsupervised person re-ID methods use the k-reciprocal re-rank distance (Zhong et al., 2017), which is a weighted combination of Euclidean distance and Jaccard distance between reciprocal-neighbor sets, calculated over the entire dataset. More recently, STML (Kim et al., 2022) proposes an unsupervised metric learning framework for image retrieval using a simpler batch-wise contextual similarity measure. We loosely follow STML's contextual similarity definition, making significant changes to accommodate the change in problem setting and to address optimization issues (these changes are enumerated in Appendix E.1). We emphasize that prior work on contextual similarity *optimizes the cosine similarity* towards the contextual similarity, focusing on the unsupervised scenario, while our work *optimizes the contextual similarity* towards the true similarity, requiring full supervision.

Robust Metric Learning Over-reliance on binary supervision is a long-standing problem in metric learning. Many studies overcome this issue by taking advantage of the hierarchical nature of labels in metric learning datasets. Sun et al. (2021), Zheng et al. (2022), and Ramzi et al. (2022) explicitly use hierarchical labels for training. These methods assign a higher cost to mistakes in discriminating labels that are farther apart in the hierarchy, leading to a more robust embedding space. Yan et al. (2021) propose to generate synthetic hierarchical labels for unsupervised metric learning, and Yan et al. (2023) extend this idea to metric learning with synthetic label noise. These two works use a hyperbolic embedding space to better capture hierarchical relationships (Khrulkov et al., 2020). Ermolov et al. (2022) show that simply using a hyperbolic embedding space instead of a Euclidean embedding space improves metric learning performance. Our work has a similar motivation to the above hierarchical and hyperbolic metric learning works, but we use contextual similarity instead of hierarchical labels to mitigate label inconsistency. Appendix I.4 contains some results on hierarchical retrieval metrics.

3. Method

Notation Denote the normalized output of the embedding network as $f_i \in \mathbb{R}^d$. $s_{ij} = \langle f_i, f_j \rangle \in [-1, 1]$ denotes the cosine similarity between the samples i and j. There are n samples in a mini-batch. We always use balanced sampling, where k images are selected from n/k randomly sampled labels. n is divisible by k. k is divisible by 2, but we always use $k \geq 4$ in experiments. $y_{ij} \in \{0, 1\}$ denotes the true similarity between i and j, defined as $y_{ij} = 1$ if samples i and j share the same label and 0 otherwise. We use uppercase letters to denote matrices, math script to denote sets, and lowercase letters to denote scalars. i, j and p are reserved for sample indices. $\mathbb{1}_{\mathcal{N}}$ is used to denote the binary indicator matrix for set $\mathcal{N}$. For instance, let $\mathcal{N}(i)$ denote the set of neighbors to sample i, then $\mathbb{1}_{\mathcal{N}}(i, j) = 1$ if $j \in \mathcal{N}(i)$, and 0 otherwise.

Contextual Similarity Definition We loosely follow the definition of contextual similarity proposed in STML (Kim et al., 2022), with significant modifications to accommodate the change in problem setting and to address optimization issues (these modifications are enumerated in Appendix E.1). In this section, we present the similarity definition using indicator matrices in order to show an efficient implementation in PyTorch. Note that the binary "and" is replaced by multiplication for differentiability. Algorithm 1 contains PyTorch-like pseudo-code for Equations 1 - 7. We include the code here for reproducibility and to show that our contextual loss can be compactly implemented despite the cumbersome mathematical notation.

We denote the contextual similarity between samples i and j as w_{ij}. The matrix with entries w_{ij} is entirely a function of the cosine similarity matrix with entries s_{ij}. The goal of Equations 1 - 4 is to calculate w_{ij} in terms of s_{ij}. This is implemented as `get_contextual_similarity` in Algorithm 1. For readability, we present the w_{ij} calculation as

Algorithm 1 Pseudo-code, PyTorch-like

```
# Hyperparameters: alpha, k, eps, s_tilde, lam, gamma
# The symbol '@' means matrix multiplication in Python
# Note: In PyTorch, set keepdim=True when calling sum(.)

class GreaterThan(autograd.Function):
    # Implements theta with heuristic gradient
    def forward(x, y):
        return (x >= y).float()
    def backward(g): # Returns gradient w.r.t (x, y)
        return g * alpha, - g * alpha

def get_contextual_similarity(s, k, eps):
    D = 2 - 2 * s                       # Squared Euclidean distance
    Dk = -(-D).topk(k).values[:,-1:] # Distance to k-th neighbor

    Nk_mask = GreaterThan(-D + eps, -Dk.detach())
    M_plus = (Nk_mask @ Nk_mask.T) / Nk_mask.sum(dim=1).detach()
    Nk_mask_not = 1 - Nk_mask
    M_minus = (Nk_mask_not @ Nk_mask_not.T)
                                / Nk_mask_not.sum(dim=1).detach()
    W_1 = 0.5 * (M_plus + M_minus) * Nk_mask

    # Distance to k/2-th neighbor
    Dk_over_2 = -(-D).topk(k//2).values[:,-1:]
    Nk_over_2_mask = GreaterThan(-D + eps, -Dk_over_2.detach())
    Rk_over_2_mask = Nk_over_2_mask * Nk_over_2_mask.T
    W_2 = (Rk_over_2_mask @ W_1) / Rk_over_2_mask.sum(dim=1)

    return 0.5 * (W_2 + W_2.T)

for data, labels in loader:
    f = F.normalize(model(data)) # normalized embeddings
    s = f @ f.T                  # cosine similarity matrix
    y = (labels.T == labels)     # true similarity matrix
    w = get_contextual_similarity(s, k, eps) # matrix

    I_neg = 1 - eye(w.shape[0])  # ones with zeros on diagonal
    L_contrast = contrastive(s, y) # Standard, code omitted
    L_reg = (s.mean() - s_tilde).square()
    L_context = ((w - s).square()* I_neg).mean()
    loss = lam * L_context + (1-lam) * L_contrast + gamma * L_reg
    loss.backward()
    optimizer.step()
```

three sequential steps.

Step 1 Neighborhood Calculation The first step calculates a binary matrix $\mathbb{1}_{\mathcal{N}_{k+\epsilon}}(i,j)$ indicating whether sample j is a neighbor of i. This binary value can be thought of as a preliminary prediction of y_{ij}. The neighborhood indicator calculation can be defined in terms of the heaviside function, which has no gradient. We set a constant positive gradient in the backward pass, which is reasonable since θ is a (non-strictly) increasing function.

$$\begin{aligned} &\text{Forward: } \theta(x) = 1, \text{ if } x \geq 0\,;\ 0 \text{ otherwise.} \\ &\text{Backward: } \frac{\partial \theta(x)}{\partial x} = \alpha. \end{aligned} \tag{1}$$

Let $D(i,j)$ denote the squared Euclidean distance between samples i and j. By definition, $D(i,j) = 2 - 2s_{ij}$ and $D(i,j) \in [0,4]$. The sg operator denotes stop gradient.

Using θ, we calculate the indicator function for whether sample j is in the $k+\epsilon$ neighborhood of sample i:

$$\begin{aligned} &\mathbb{1}_{\mathcal{N}_{k+\epsilon}}(i,j) = \theta(-D(i,j) + \text{sg}(D(i,p)) + \epsilon), \\ &\text{where } p \text{ denotes the } k\text{-th closest neighbor of } i. \end{aligned} \tag{2}$$

In words, $\mathbb{1}_{\mathcal{N}_{k+\epsilon}}(i,j)$ is a binary value indicating whether or not $D(i,j) \leq D(i,p) + \epsilon$. By convention, the sample itself is always included in the closest neighbor count (e.g. if $k = 2$, then the "k-th closest neighbor" is the closest neighbor to a sample). We now proceed to calculate the intersection of neighborhood sets.

Step 2 Intersection of Neighborhoods This step refines the similarity prediction by counting the number of neighbors two samples have in common. Intuitively, samples with the same label should have a similar set of neighbors.

$$\begin{aligned} &W_1(i,j) = \frac{\mathbb{1}_{\mathcal{N}_{k+\epsilon}}(i,j)}{2} \cdot \\ &\qquad \left(\frac{M_+(i,j)}{\text{sg} \sum_p \mathbb{1}_{\mathcal{N}_{k+\epsilon}}(i,p)} + \frac{M_-(i,j)}{\text{sg} \sum_p 1 - \mathbb{1}_{\mathcal{N}_{k+\epsilon}}(i,p)} \right), \\ &\text{where } M_+(i,j) = \sum_{p=1}^{n} \mathbb{1}_{\mathcal{N}_{k+\epsilon}}(i,p) \mathbb{1}_{\mathcal{N}_{k+\epsilon}}(j,p), \\ &\text{and } M_-(i,j) = \sum_{p=1}^{n} \left(1 - \mathbb{1}_{\mathcal{N}_{k+\epsilon}}(i,p)\right) \left(1 - \mathbb{1}_{\mathcal{N}_{k+\epsilon}}(j,p)\right). \end{aligned} \tag{3}$$

$W_1(i,j) \in [0,1]$ is an intermediary similarity value. $M_+(i,j)$ counts the number of neighbors i and j have in common. $M_-(i,j)$ counts the number of non-neighbors i and j have in common. Appendix E.3 Figure 12 explains why both M_+ and M_- are necessary. The normalization factors in Eq. 3 ensure that the similarity value is between 0 and 1. We do not backpropagate gradients through the normalization factors, because it is undesirable to optimize the number of samples in the neighborhood set. As further justification for the stop gradient, note that $\sum_p \mathbb{1}_{\mathcal{N}_{k+\epsilon}}(i,p) = k$ for any i, p when $\epsilon = 0$.

Step 3 Query Expansion This final step further refines the similarity prediction by averaging W_1 across close neighbors (known as query expansion, see Arandjelović & Zisserman (2012)).

$$\begin{aligned} \mathbb{1}_{\mathcal{R}_{k/2+\epsilon}}(i,j) &= \mathbb{1}_{\mathcal{N}_{k/2+\epsilon}}(i,j) \mathbb{1}_{\mathcal{N}_{k/2+\epsilon}}(j,i) \\ W_2(i,j) &= \frac{\sum_p \mathbb{1}_{\mathcal{R}_{k/2+\epsilon}}(i,p) W_1(p,j)}{\sum_p \mathbb{1}_{\mathcal{R}_{k/2+\epsilon}}(i,p)} \\ w_{ij} &= \frac{1}{2}\left(W_2(i,j) + W_2(j,i)\right) \end{aligned} \tag{4}$$

$\mathbb{1}_{\mathcal{R}_{k/2+\epsilon}}(i,j)$ is a binary value which equals 1 if j is a $k/2+\epsilon$ neighbor of i and i is a $k/2+\epsilon$ neighbor of j, 0 otherwise. This type of reciprocal relationship is widely used in the retrieval literature, most notably by Zhong et al. (2017). $W_2(i,j)$ is an intermediary similarity value representing the entries of W_1 averaged over the smaller $\mathcal{R}_{k/2+\epsilon}$ neighborhood. W_2 is then symmeterized to yield the final contextual similarity values $w_{ij} \in [0,1]$.

Loss Function We use the MSE loss to optimize w_{ij} against

the true similarity labels y_{ij}:

$$\mathcal{L}_{\text{context}} = \frac{1}{n^2} \sum_{i,j|i\neq j} (y_{ij} - w_{ij})^2 \tag{5}$$

Our final loss function $\mathcal{L}_{\text{ours}}$ is a sum of three loss functions:

$$\mathcal{L}_{\text{ours}} = \lambda \mathcal{L}_{\text{context}} + (1-\lambda)\mathcal{L}_{\text{contrast}} + \gamma \mathcal{L}_{\text{reg}} \tag{6}$$

$$\mathcal{L}_{\text{reg}} = \left(\tilde{s} - \frac{1}{n^2} \sum_{i,j}^{n^2} s_{ij} \right)^2 \tag{7}$$

$\mathcal{L}_{\text{contrast}}$ is the standard contrastive loss (see Appendix E.3 Eq. 29). In our work, $\mathcal{L}_{\text{contrast}}$ is best viewed as a regularizer that reduces the decomposability gap between the batch-wise contextual loss and the contextual loss over the entire dataset. We justify this interpretation in Appendix D Fig. 9. $\mathcal{L}_{\text{reg}}$ is a similarity regularizer that encourages the model to use the entire embedding space by pushing the average cosine similarity between all pairs towards the constant $\tilde{s}$.

Remarks The parameter w_{ij} is a function of s_{ij}, so all three components of our loss function in Eq. 6 optimize the cosine similarity matrix with entries s_{ij}. However $\mathcal{L}_{\text{context}}$ is the main contribution of the current work, and experiments verify that most of the improvement over baselines can be attributed to this contextual loss. The value of k is not arbitrary; it must be set to the number of samples per label in the mini-batch. Although the time and space to calculate the contextual loss scales as $O(n^3)$, all operations are implemented as matrix multiplication, which is highly optimized on modern hardware. Appendix C Figure 8 shows that the cubic scaling is negligible for all practical batch sizes.

Hyperparameters α controls the magnitude of the heaviside gradient. Tuning α is unnecessary, since it is redundant with the learning rate. ϵ is the desired similarity margin between positive and negative samples. ϵ is analogous to the margin parameter on the triplet and multi-similarity loss. δ_+ and δ_- (Appendix E.3 Eq. 29) are the positive and negative margins resp. for the contrastive loss. $\tilde{s}$ is the desired average cosine similarity between all pairs. λ and γ control the relative weighting between the three losses. The choice of $(1-\lambda)$ for the weight on the contrastive loss instead of a separate hyperparameter is completely arbitrary, as tuning the contrastive loss weight separately would be redundant with tuning the learning rate.

4. Analysis

This section discusses intuition behind the contextual loss in Eq. 5. Section 4.1 provides empirical evidence that $\mathcal{L}_{\text{context}}$ converges and shows that $\mathcal{L}_{\text{context}} = 0$ coincides with the correct ranking of samples. Sections 4.2 - 4.4 carefully justify the *semantic consistency* argument outlined in the introduction.

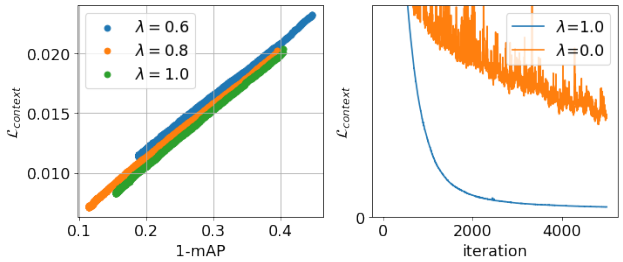

Figure 3. Left: plot of contextual loss value vs. 1-mAP (mean AP) over the course of training on CUB for different choices of λ. Training proceeds from upper right to bottom left. Observe that 1-mAP decreases as contextual loss decreases. This shows that $\mathcal{L}_{\text{context}}$ is a valid surrogate for learning to rank. Right: convergence plot of $\mathcal{L}_{\text{context}}$ on CUB without mini-batching. $\mathcal{L}_{\text{context}}$ decreases almost monotonically when the contextual loss is minimized ($\lambda = 1$), while there is a large amount of noise when the contrastive loss is minimized ($\lambda = 0$). $\gamma = 0$.

4.1. Contextual Loss and Optimization

Proposition 4.1. *For a batch of size n with exactly k samples from each class (n divisible by k, $n > 2k$, and $k \geq 2$), assuming that $\epsilon = 0$, $\mathcal{L}_{context} = 0$ if and only if all samples are correctly ranked with respect to every other sample within the batch, i.e. $s_{ip} > s_{ij}$, $\forall p, j$ where $y_{ij} = 0$ and $y_{ip} = 1$, $\forall i \in [1, n]$.*

We defer the proof to Appendix B. This Proposition shows that $\mathcal{L}_{\text{context}}$ is a valid ranking objective, similar to AP surrogates, multi-similarity, and triplet losses. Note that Proposition 4.1 does not hold for $\mathcal{L}_{\text{contrast}}$, since $\mathcal{L}_{\text{contrast}}$ continues to provide gradients up to fixed margins, regardless of whether the correct ranking is satisfied. Figure 3 (left) shows that $\mathcal{L}_{\text{context}}$ is approximately a linearly scaled version of 1-mAP over the course of training. Figure 3 (right) suggests that the value of $\mathcal{L}_{\text{context}}$ converges when optimized using gradient descent. The Appendix contains more empirical evidence that the value of $\mathcal{L}_{\text{context}}$ converges (Fig. 9 and 15). Figure 4 justifies the choice of heuristic gradient in Eq. 1. In this simple 2-D example, the gradient is always positive and non-zero in the direction away from the minimum, until the minimum is reached.

4.2. Intuition

In the previous subsection, we proved that the minimum of $\mathcal{L}_{\text{context}}$ corresponds to a correct ranking of samples within a batch. We also showed that the value of $\mathcal{L}_{\text{context}}$ converges empirically. However, we still need some intuition as to why gradients from $\mathcal{L}_{\text{context}}$ work better than simple pair-wise contrastive loss functions. This discussion will naturally lead to the *semantic consistency* intuition promised at the beginning of the paper. Let us start by asking: *what is the value of optimizing the intersection of neighborhood sets in*

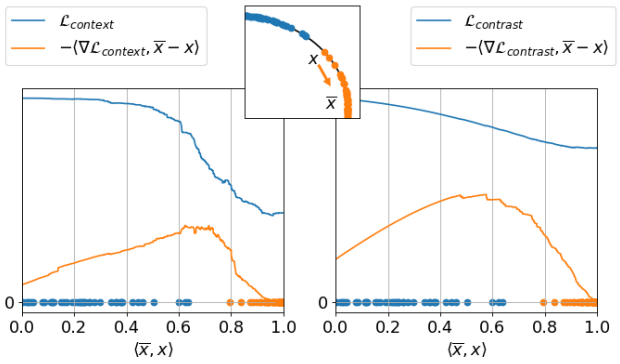

Figure 4. Illustration of $\mathcal{L}_{\text{context}}$ and its gradient. We generate 64 random points on the unit circle from two classes, centered around the coordinates (0,1) and (1,0) (see middle plot). We move one orange point x from the blue centroid to the orange centroid $\overline{x}$. The value of the loss function is plotted in blue, and the amount of gradient pointing away from $\overline{x}$ is plotted in orange. $\mathcal{L}_{\text{context}}$ decreases as x rolls to the right, and the gradient closely approximates what it should look like if $\mathcal{L}_{\text{context}}$ were continuous. We include the same illustration for $\mathcal{L}_{\text{contrast}}$ on the right for comparison.

the manner of Eq. 3? We offer a straight-forward intuition: *Maximizing the intersection between the neighborhood sets of two samples is equivalent to pushing one sample towards the context of the other sample and vice versa*; *minimizing the intersection is equivalent to pulling apart the contexts of the two samples*. We show this intuition by analyzing the gradient.

For simplicity, consider $\epsilon = 0$, such that the normalization factors in Eq. 3 are constant: $\sum_p \mathbb{1}_{\mathcal{N}_{k+\epsilon}}(i,p) = k$ and $\sum_p 1 - \mathbb{1}_{\mathcal{N}_{k+\epsilon}}(i,p) = n - k$. For the remainder of the section we drop the $k+\epsilon$ subscript from $\mathcal{N}$ for readability. Further consider a negative pair of samples i and j ($y_{ij} = 0$), where j is wrongly ranked w.r.t. i (i.e. $\mathbb{1}_{\mathcal{N}}(i,j) = 1$). For the sake of developing intuition, let us further assume that all entries of W_1 are correct except index i,j; also, $\mathbb{1}_{\mathcal{R}_{k/2+\epsilon}}(i,p) = 0\,\forall p \neq i$. Under these assumptions, $w_{ij} = W_1(i,j)$ and $\mathcal{L}_{\text{context}} = \frac{1}{2n^2}(y_{ij} - w_{ij})^2$. This allows us to focus on interpreting Eq. 3. Under these assumptions, Eq. 3 simplifies to (see algebra in Appendix A):

$$W_1(i,j) = \underbrace{\mathbb{1}_{\mathcal{N}}(i,j)}_{①}\underbrace{(a\langle\mathbb{1}_{\mathcal{N}}(i), \mathbb{1}_{\mathcal{N}}(j)\rangle + b)}_{②}, \quad \text{where } a = \frac{1}{2k} + \frac{1}{2(n-k)}, \text{ and } b = \frac{n-2k}{2(n-k)}. \tag{8}$$

a and b are positive constants, assuming that $n > 2k$. $\langle\cdot\rangle$ denotes the inner product between the i-th and j-th rows of the indicator matrix. This inner product is clearly positive. $W_1(i,j) \in (0,1]$ must be a non-zero positive number under our assumptions. The gradient w.r.t. $W_1(i,j)$ must be non-zero positive because $y_{ij} = 0$. We are now ready for the backward pass. For simplicity of notation, $\partial W_1(i,j) := g_{ij} > 0$ denotes the gradient of the loss w.r.t. $W_1(i,j)$. The gradient w.r.t. the distance matrix ∂D can be split into two parts ① and ②, added together by chain rule.

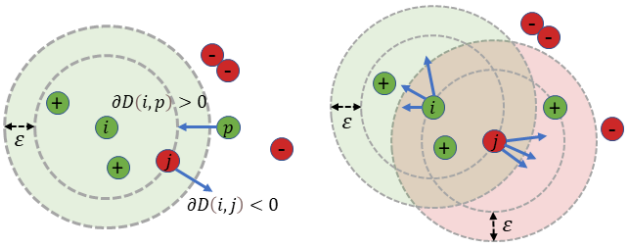

Figure 5. This figure complements Eq. 9. Part ① of the gradient (left) enforces correct ranking w.r.t. sample i. Part ② of the gradient (right) increases the distance between i and all samples in the neighborhood of j, and vice versa. As shown, sometimes this implies that samples with the same label are pulled apart. The $k+\epsilon$ neighborhood as defined in Eq. 2 is shaded. $k = 4$.

$$\begin{aligned} &① \; \partial D(i,j) = -\alpha\partial\mathbb{1}_{\mathcal{N}}(i,j) = -\alpha g_{ij}\,(a\langle\cdot\rangle + b) \\ &② \begin{cases} \partial D(i) = -\alpha\partial\mathbb{1}_{\mathcal{N}}(i) & = -\alpha a g_{ij}\mathbb{1}_{\mathcal{N}}(i,j)\mathbb{1}_{\mathcal{N}}(j) \\ \partial D(j) = -\alpha\partial\mathbb{1}_{\mathcal{N}}(j) & = -\alpha a g_{ij}\mathbb{1}_{\mathcal{N}}(i,j)\mathbb{1}_{\mathcal{N}}(i) \end{cases} \end{aligned} \tag{9}$$

Note that the $-\alpha$ factors in Eq. 9 come from going backward through $\theta(\cdot)$. The negative sign accounts for optimizing distance instead of similarity. Intuitively, the two components of the gradient perform different functions. The gradient in Eq. 9 ① enforces correct ranking. The gradient in Eq. 9 ② pulls i away from the context of j and j away from the context of i. More clearly, $\partial D(i,p) < 0$ when $\mathbb{1}_{\mathcal{N}}(j,p) = 1$ and $\partial D(i,p) = 0$ otherwise, for all samples p in the batch. In words, we increase the distance between i and all samples in *the neighborhood* of j. See Figure 5 for an illustration.

4.3. Semantic Consistency

The previous subsection showed that the gradients of the contextual loss optimize distances between neighbors of samples, not just pairwise distances. This is important because the neighborhood of a sample contains semantically similar images, regardless of whether they have the same label or not. Indeed, optimizing contextual similarity may result in pulling apart samples with the *same label*, and to a lesser extent pushing together samples with *different labels*. For example, in Figure 5, sample i is pulled both from sample j, which has a different label, and from the two neighbors of sample j, which have the same labels as sample i.

This behavior is novel to the contextual loss. All of our baselines which take the cosine similarity matrix as input satisfy the condition that $\partial D(i,j) \geq 0$ when $y_{ij} = 1$ and

Table 1. Comparison of train and test accuracy on CUB between $\mathcal{L}_{\text{context}}$ and $\mathcal{L}_{\text{context}}$ with gradient corrected according to Eq. 10.

CUB	$\mathcal{L}_{\text{context}}$	with gradient correction
Train R@1	87.0	92.9
Test R@1	71.4	65.4

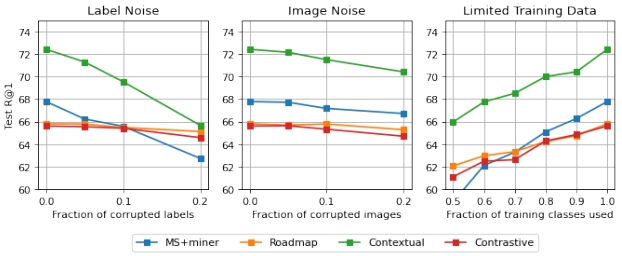

Figure 6. Robustness and generalizability comparison of the contextual loss against other pairwise ranking losses. See description in Section 4.4. For this experiment, we test $\mathcal{L}_{\text{context}}$ without additional regularization, i.e. $\lambda = 1$ and $\gamma = 0$.

$\partial D(i,j) \leq 0$ when $y_{ij} = 0$. Note that this discussion is limited to pairwise ranking losses; classification methods cannot be analyzed in this way. We analyze the effect of these "wrong" gradients in Table 1 by adding a custom autograd function between the distance matrix and the contextual loss. This function truncates the gradient of the distance matrix according to the label matrix:

$$\partial D(i,j) = \begin{cases} \max(\partial D(i,j), 0), & \text{if } y_{ij} = 1 \\ \min(\partial D(i,j), 0), & \text{otherwise.} \end{cases} \tag{10}$$

Clearly, clamping the distance gradients according to Eq. 10 raises the R@1 on training data at the expense of a lower test R@1. This raises the question: *why does discarding seemingly wrong gradients lower the R@1 accuracy by 6%?* We hypothesize that $\mathcal{L}_{\text{context}}$ implicitly regularizes the embedding space against the label and image noise illustrated in Fig. 1. The contextual loss considers relationships between groups of k samples, which intuitively should be more robust to random label variations than solely relying on pairwise relationships. In other words, sample pairs with gradients $\partial D(i,j)$ that violate the true labels y_{ij} are pairs where the y_{ij} is inconsistent with semantic information; these labels likely do not generalize to test data. We justify this hypothesis in the next section by testing the robustness and generalizability of our approach.

4.4. Robustness and Generalizability Experiments

The previous subsection proposed that optimizing contextual similarity with $\mathcal{L}_{\text{context}}$ leads to an embedding space with higher test R@1 accuracy because contextual similarity is more robust to label noise and more generalizable to test data. We verify this claim with three sets of experiments on the CUB benchmark in Figure 6.

Label Noise We experiment with assigning random labels to 5, 10, and 20 % of randomly selected training samples.

Image Noise We scraped around 1,800 generic bird images from Bing using search terms such as "bird" and "flying bird". We manually filtered the images such that each image contains at least one bird. We then replace a percentage of randomly selected CUB training samples with random images from our Bing dataset. This experiment simulates a typical web-scraped dataset, where images are often only lightly proofread by a human. In most modern image datasets, a small portion of labels are either wrong or do not accurately reflect the content of the image. We experiment with 5, 10, and 20 % image noise.

Limited Training Data According to traditional machine learning wisdom, small datasets are easier to overfit. We demonstrate that our method achieves reasonable results even when some training data is withheld. CUB-200 is already the smallest standard benchmark, with 5,994 training samples belonging to 100 classes. We experiment with using only the first 50, 60, 70, 80, and 90 classes for training.

The results from these three sets of experiments are presented in Fig. 6. Our method clearly outperforms other pairwise ranking losses. We tune learning rates individually for each method. We use batch size 256 and $k = 8$. We use $\mathcal{L}_{\text{context}}$ by itself instead of the entire loss to study the contextual loss without additional regularization.

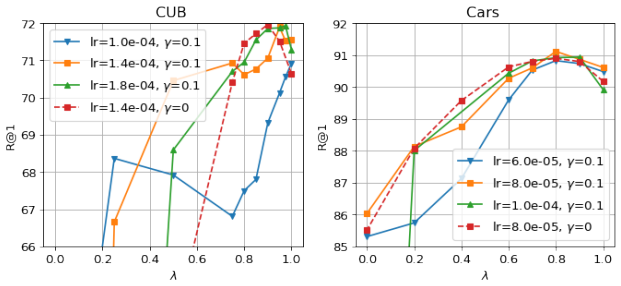

Figure 7. Ablation results. We test the contribution of each component in Eq. 6 by trying different values for λ and γ on CUB and Cars. The dashed line indicates results without the similarity regularizer ($\gamma = 0$). The three different marker symbols represent different learning rates. We show results for different learning rates because the optimal learning rate varies with λ. Observe that the optimal R@1 is always achieved by a combination of $\mathcal{L}_{\text{contrast}}$ and $\mathcal{L}_{\text{context}}$. The similarity regularizer is sometimes helpful, but never detrimental.

5. Experiments

Datasets We experiment on two small-scale and two large-scale datasets: Caltech-UCSD Birds (CUB-200) (Wah et al.,

Table 2. State-of-the-art comparisons on CUB and Cars datasets. We use ResNet-50. The last two rows use embedding size 1536, while the other rows use embedding size 512. Results should not be compared across embedding sizes. † indicates our reproduction using the implementation by Musgrave et al. (2020b); other results are copied from their original paper. All results indicated by † are an average of 6 trials with different random seeds but with the same train-test split.

	CUB					Cars				
Method	R@1	R@2	R@4	R@8	mAP	R@1	R@2	R@4	R@8	mAP
DRML (Zheng et al., 2021)	68.7	78.6	86.3	91.6	-	86.9	92.1	95.2	97.4	-
DIML (Zhao et al., 2021)	68.2	-	-	-	-	87.0	-	-	-	-
DiVA (Milbich et al., 2020)	69.2	79.3	-	-	-	87.6	92.9	-	-	-
Proxy Anchor (Kim et al., 2020)	69.7	80.0	87.0	92.4	-	87.7	92.9	95.8	97.9	-
MS (Wang et al., 2019)	67.8	77.8	85.6	-	-	87.8	92.7	95.3	-	-
IBC (Seidenschwarz et al., 2021)	70.3	80.3	87.6	-	-	88.1	93.3	96.2	-	-
S2SD (Roth et al., 2020)	70.1	79.7	-	-	-	89.5	93.9	-	-	-
Proxy NCA + Metrix	70.4	80.6	**88.7**	-	-	88.5	93.4	96.5	-	-
PA + Metrix	71.0	**81.8**	88.2	-	-	89.1	93.6	96.7	-	-
MS + Metrix (Venkataramanan et al., 2021)	71.4	80.6	86.8	-	-	89.6	94.2	96.0	-	-
HIST (Lim et al., 2022)	71.4	81.1	88.1	-	-	89.6	93.9	96.4	-	-
MHGL (Ebrahimpour et al., 2022)	70.6	80.9	88.0	92.3	-	90.1	94.2	96.4	98.1	-
MS † (Wang et al., 2019)	65.9	76.6	84.9	90.8	36.1	82.1	88.6	92.9	95.7	35.7
Contrastive † (Hadsell et al., 2006)	65.9	76.6	84.8	90.8	35.3	82.4	88.7	92.8	95.6	35.3
Roadmap † (Ramzi et al., 2021)	66.0	76.8	85.3	91.1	36.0	83.5	89.8	93.7	96.3	37.3
Triplet † (Weinberger et al., 2005)	64.8	75.9	84.5	90.6	33.8	88.1	93.1	96.0	97.7	41.8
MS + miner † (Wang et al., 2019)	68.0	78.4	86.2	91.7	36.4	90.5	94.7	97.0	**98.4**	42.7
Proxy Anchor † (Kim et al., 2020)	69.1	79.5	87.0	92.2	37.4	89.0	93.5	96.2	97.9	38.8
Proxy NCA † (Teh et al., 2020)	66.3	77.1	85.5	91.5	35.1	88.2	93.2	96.1	97.9	38.2
Contextual (Ours) †	**71.9**	81.5	88.5	**93.1**	**40.2**	**91.1**	**95.0**	**97.1**	**98.4**	**43.3**
PA + AVSL (Zhang et al., 2022)	71.9	81.7	88.1	93.2	-	91.5	95.0	97.0	98.4	-
Contextual (Ours) †	**72.7**	**82.2**	**88.8**	**93.4**	**40.5**	**91.8**	**95.4**	**97.4**	**98.6**	**43.3**

Table 3. State-of-the-art comparisons on SOP. We use ResNet-50. The last two rows use embedding size 1536, while the other rows use 512. All results indicated by † are an average of 2 trials with different random seeds but with the same train-test split.

Method	R@1	R@10	R@100	R@1000
DRML	79.9	90.7	96.1	-
DIML	79.3	-	-	-
DiVA	79.6	91.2	-	-
Proxy Anchor	79.1	90.8	96.2	-
MS	76.9	89.8	95.9	-
IBC	81.4	91.3	95.9	-
S2SD	80.0	91.4	-	-
Proxy NCA + Metrix	81.3	**92.7**	97.1	-
PA + Metrix	81.3	91.7	96.9	-
MS + Metrix	81.0	92.0	**97.2**	-
HIST	81.4	92.0	96.7	-
MHGL	81.7	92.0	96.6	-
MS †	79.9	90.4	95.8	98.6
Contrastive †	80.9	90.9	95.6	98.3
Roadmap †	81.9	92.0	96.3	98.7
Triplet †	81.9	92.5	96.8	**98.9**
MS + miner †	82.2	92.5	96.7	98.8
Proxy Anchor †	79.7	91.1	96.2	98.7
Proxy NCA †	78.8	90.7	96.3	98.8
Contextual (Ours) †	**82.6**	92.5	96.7	98.8
PA + AVSL	79.6	91.4	96.4	-
Contextual (Ours) †	**83.2**	**92.9**	**96.8**	**98.8**

2011), Stanford Cars-196 (Krause et al., 2013), Stanford Online Products (SOP) (Oh Song et al., 2016), and mini-iNaturalist-2021 (Van Horn et al., 2018). CUB-200 and Cars-196 are smaller fine-grain classification datasets with 200 and 196 unique labels, respectively. SOP is a large-scale dataset with 120,053 product images from 22,634 classes. mini-iNaturalist-2021 is a subset of the iNaturalist-2021 species classification competition dataset, with 50 images from each of 10,000 species. iNaturalist results are deferred to Appendix G Table 4.

Baselines We compare against a diverse set of baselines in Tables 2 and 3. Some baselines (such as DRML, Metrix, S2SD, HIST, MHGL, and AVSL) use complicated tricks to achieve published results. We simply copy the results for these baselines from their original paper. We then choose a representative set of baselines which only modify the loss function, and reproduce their results under identical experimental conditions (indicated by † in the tables). We first compare against contrastive and triplet losses, which are the accepted standard in the field. Multi-similarity (MS) is a popular pairwise ranking loss. From classification methods, we compare against proxy anchor and proxy NCA. From AP maximization methods, we compare against Fast-AP, Smooth-AP, and Roadmap (Fast-AP and Smooth-AP comparisons are deferred to Appendix G Table 4). The benchmark results for many of the above-mentioned loss functions

are under-represented in the literature. For fair comparison, we tune learning rates separately for each loss function. We use default values for any other hyperparameters.

Hyperparameters and Setup On our method, we tune λ and ϵ separately for each dataset. We use fixed values for remaining hyperparameters: $\gamma = 0.1$, $\alpha = 10.0$, $k = 4$, $\delta_+ = 0.75$, $\delta_- = 0.6$, $\tilde{s} = 0.3$. The results in the main paper all use 224×224 image resolution. Some recent studies use 256×256 image resolution; comparisons in this setting are included in Appendix G Table 4. We use Adam with a decaying learning-rate schedule. We report results on the model with the best test R@1 metric, as is standard in the literature. We tune learning rates separately for each method and dataset combination. We use a batch size of 256 for iNaturalist, 128 for SOP and CUB, and 64 for Cars; the larger batch size is necessary to achieve reasonable performance on iNaturalist, while the smaller batch size appears to reduce overfitting on Cars. We use a 4 per class balanced sampler. For SOP and iNaturalist, we use hierarchical sampling (Cakir et al., 2019), following prior work. We use an embedding size of 512 for most comparisons; we only use an embedding size of 1536 for comparison with AVSL. We use ResNet-50 with a linear embedding layer. For CUB and Cars, we add an additional linear projector layer, which is discarded at the end of training. We use GeM pooling (Radenović et al., 2018), a widely used generalization of max and mean pooling. We always freeze batch-norm. We emphasize that the setup described above is used for all baseline results marked by †. Additionally, we add the linear projector and/or similarity regularization with $\gamma = 0.1$ only when it improves the baseline, for fair comparison.

Performance Metrics We report Recall @ k for select k. R@k is the percentage of test samples where at least one of the k closest neighbors have the same label. We also report mAP, a standard ranking metric defined in Appendix I.3. We report the average of 6 trials on CUB and Cars, and 2 trials on SOP. We omit standard deviations for readability.

Discussion Our R@1 results are better than the best baseline across all datasets and embedding sizes. We achieve R@1 gains of 0.5%, 0.6 %, and 0.4 % on CUB, Cars, and SOP resp. for the 512 embedding size. We achieve gains of 0.8%, 0.3%, and 3.6% on these datasets for the 1536 embedding size. These results are an average of 6 trials with different random seeds on CUB and Cars, and 2 trials on SOP. Additionally, we achieve R@1 gains of 0.8%, 0.6%, 0.2%, and 0.3% over the best baseline on CUB, Cars, SOP, and iNaturalist, resp. averaged over 3 trials, under slightly different experimental settings using 256×256 image resolution (see Table 4 in the appendix).

Ablation Figure 7 plots R@1 on CUB and Cars with different values of λ and γ. This figure shows that the best R@1 performance is always achieved by a combination of contextual and contrastive losses. The optimal value of λ is usually between 0.8 and 0.9. The similarity regularizer is only needed to achieve state-of-the-art on some datasets. This property is reasonable for a regularizer, and it is more important to observe that the similarity regularizer improves the results of many diverse metric learning losses (see Table 6 Appendix I.1).

6. Conclusion

In this work, we proposed a novel contextual loss based on contextual similarity optimization. Our contextual loss improves the R@1 performance significantly over the current state-of-the-art across four benchmarks, when regularized by the contrastive loss and a novel similarity regularizer. We established that our loss function reduces overfitting by regularizing the embedding space for *semantic consistency among neighbors*. We justified this interpretation both analytically by inspecting the gradient, and empirically by showing that our loss function is more robust to label noise. We carefully supported each component of our loss function through extensive experiments across two different experimental settings, accompanied by exhaustive ablation studies.

Ethics Statement

We note that metric learning can be applied to controversial problems such as person re-identification and face re-identification. Our work is mainly foundational, so does not contribute directly to these applications. We also limit our experimentation to the image retrieval aspect of metric learning.

Reproducibility Statement

Instructions on how to run the code is provided in a README file. We include details on hardware requirements in Appendix H.1. Code is released here: https://github.com/Chris210634/metric-learning-using-contextual-similarity

Acknowledgments

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited. This material is based upon work supported by the Under Secretary of Defense for Research and Engineering under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Under Secretary of Defense for Research and Engineering.

References

Arandjelović, R. and Zisserman, A. Three things everyone should know to improve object retrieval. In *2012 IEEE conference on computer vision and pattern recognition*, pp. 2911–2918. IEEE, 2012.

Brown, A., Xie, W., Kalogeiton, V., and Zisserman, A. Smooth-ap: Smoothing the path towards large-scale image retrieval. In *European Conference on Computer Vision*, pp. 677–694. Springer, 2020.

Cakir, F., He, K., Xia, X., Kulis, B., and Sclaroff, S. Deep metric learning to rank. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1861–1870, 2019.

Cao, B., Araujo, A., and Sim, J. Unifying deep local and global features for image search. In *European Conference on Computer Vision*, pp. 726–743. Springer, 2020.

Chang, D., Pang, K., Zheng, Y., Ma, Z., Song, Y.-Z., and Guo, J. Your" flamingo" is my" bird": fine-grained, or not. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11476–11485, 2021.

Cho, K., Van Merriënboer, B., Bahdanau, D., and Bengio, Y. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

Chu, R., Sun, Y., Li, Y., Liu, Z., Zhang, C., and Wei, Y. Vehicle re-identification with viewpoint-aware metric learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 8282–8291, 2019.

Deng, J., Guo, J., Xue, N., and Zafeiriou, S. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4690–4699, 2019.

Ebrahimpour, M. K., Qian, G., and Beach, A. Multi-head deep metric learning using global and local representations. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 3031–3040, 2022.

El-Nouby, A., Neverova, N., Laptev, I., and Jégou, H. Training vision transformers for image retrieval. *arXiv preprint arXiv:2102.05644*, 2021.

Ermolov, A., Mirvakhabova, L., Khrulkov, V., Sebe, N., and Oseledets, I. Hyperbolic vision transformers: Combining improvements in metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7409–7419, 2022.

Guillaumin, M., Verbeek, J., and Schmid, C. Is that you? metric learning approaches for face identification. In *2009 IEEE 12th international conference on computer vision*, pp. 498–505. IEEE, 2009.

Hadsell, R., Chopra, S., and LeCun, Y. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pp. 1735–1742. IEEE, 2006.

Khrulkov, V., Mirvakhabova, L., Ustinova, E., Oseledets, I., and Lempitsky, V. Hyperbolic image embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6418–6428, 2020.

Kim, S., Kim, D., Cho, M., and Kwak, S. Proxy anchor loss for deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3238–3247, 2020.

Kim, S., Kim, D., Cho, M., and Kwak, S. Self-taught metric learning without labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7431–7441, 2022.

Krause, J., Stark, M., Deng, J., and Fei-Fei, L. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pp. 554–561, 2013.

Lim, J., Yun, S., Park, S., and Choi, J. Y. Hypergraph-induced semantic tuplet loss for deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 212–222, 2022.

Liu, Z., Luo, P., Qiu, S., Wang, X., and Tang, X. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1096–1104, 2016.

McFee, B. and Lanckriet, G. R. Metric learning to rank. In *ICML*, 2010.

Milbich, T., Roth, K., Bharadhwaj, H., Sinha, S., Bengio, Y., Ommer, B., and Cohen, J. P. Diva: Diverse visual feature aggregation for deep metric learning. In *European Conference on Computer Vision*, pp. 590–607. Springer, 2020.

Musgrave, K., Belongie, S., and Lim, S.-N. A metric learning reality check. In *European Conference on Computer Vision*, pp. 681–699. Springer, 2020a.

Musgrave, K., Belongie, S., and Lim, S.-N. Pytorch metric learning, 2020b.

Oh Song, H., Xiang, Y., Jegelka, S., and Savarese, S. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4004–4012, 2016.

Radenović, F., Tolias, G., and Chum, O. Fine-tuning cnn image retrieval with no human annotation. *IEEE transactions on pattern analysis and machine intelligence*, 41(7): 1655–1668, 2018.

Ramzi, E., Thome, N., Rambour, C., Audebert, N., and Bitot, X. Robust and decomposable average precision for image retrieval. *Advances in Neural Information Processing Systems*, 34:23569–23581, 2021.

Ramzi, E., Audebert, N., Thome, N., Rambour, C., and Bitot, X. Hierarchical average precision training for pertinent image retrieval. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XIV*, pp. 250–266. Springer, 2022.

Ranjan, V., Rasiwasia, N., and Jawahar, C. Multi-label cross-modal retrieval. In *Proceedings of the IEEE international conference on computer vision*, pp. 4094–4102, 2015.

Revaud, J., Almazán, J., Rezende, R. S., and Souza, C. R. d. Learning with average precision: Training image retrieval with a listwise loss. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5107–5116, 2019.

Rolínek, M., Musil, V., Paulus, A., Vlastelica, M., Michaelis, C., and Martius, G. Optimizing rank-based metrics with blackbox differentiation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7620–7630, 2020.

Roth, K., Milbich, T., Ommer, B., Cohen, J. P., and Ghassemi, M. S2sd: simultaneous similarity-based self-distillation for deep metric learning. *arXiv preprint arXiv:2009.08348*, 2020.

Seidenschwarz, J. D., Elezi, I., and Leal-Taixé, L. Learning intra-batch connections for deep metric learning. In *International Conference on Machine Learning*, pp. 9410–9421. PMLR, 2021.

Sun, Y., Zhu, Y., Zhang, Y., Zheng, P., Qiu, X., Zhang, C., and Wei, Y. Dynamic metric learning: Towards a scalable metric space to accommodate multiple semantic scales. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5393–5402, 2021.

Teh, E. W., DeVries, T., and Taylor, G. W. Proxynca++: Revisiting and revitalizing proxy neighborhood component analysis. In *European Conference on Computer Vision*, pp. 448–464. Springer, 2020.

Van Horn, G., Mac Aodha, O., Song, Y., Cui, Y., Sun, C., Shepard, A., Adam, H., Perona, P., and Belongie, S. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8769–8778, 2018.

Venkataramanan, S., Psomas, B., Kijak, E., Amsaleg, L., Karantzalos, K., and Avrithis, Y. It takes two to tango: Mixup for deep metric learning. *arXiv preprint arXiv:2106.04990*, 2021.

Wah, C., Branson, S., Welinder, P., Perona, P., and Belongie, S. The caltech-ucsd birds-200-2011 dataset. 2011.

Wang, X., Han, X., Huang, W., Dong, D., and Scott, M. R. Multi-similarity loss with general pair weighting for deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5022–5030, 2019.

Weinberger, K. Q., Blitzer, J., and Saul, L. Distance metric learning for large margin nearest neighbor classification. *Advances in neural information processing systems*, 18, 2005.

Weyand, T., Araujo, A., Cao, B., and Sim, J. Google landmarks dataset v2-a large-scale benchmark for instance-level recognition and retrieval. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2575–2584, 2020.

Wu, C.-Y., Manmatha, R., Smola, A. J., and Krahenbuhl, P. Sampling matters in deep embedding learning. In *Proceedings of the IEEE international conference on computer vision*, pp. 2840–2848, 2017.

Yan, J., Luo, L., Deng, C., and Huang, H. Unsupervised hyperbolic metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12465–12474, 2021.

Yan, J., Luo, L., Deng, C., and Huang, H. Adaptive hierarchical similarity metric learning with noisy labels. *IEEE Transactions on Image Processing*, 32:1245–1256, 2023.

Ye, M., Shen, J., Lin, G., Xiang, T., Shao, L., and Hoi, S. C. Deep learning for person re-identification: A survey and outlook. *IEEE transactions on pattern analysis and machine intelligence*, 44(6):2872–2893, 2021.

Zhai, A. and Wu, H.-Y. Classification is a strong baseline for deep metric learning. *arXiv preprint arXiv:1811.12649*, 2018.

Zhang, B., Zheng, W., Zhou, J., and Lu, J. Attributable visual similarity learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7532–7541, 2022.

Zhao, W., Rao, Y., Wang, Z., Lu, J., and Zhou, J. Towards interpretable deep metric learning with structural matching. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9887–9896, 2021.

Zheng, W., Zhang, B., Lu, J., and Zhou, J. Deep relational metric learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12065–12074, 2021.

Zheng, W., Huang, Y., Zhang, B., Zhou, J., and Lu, J. Dynamic metric learning with cross-level concept distillation. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXIV*, pp. 197–213. Springer, 2022.

Zhong, Z., Zheng, L., Cao, D., and Li, S. Re-ranking person re-identification with k-reciprocal encoding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1318–1327, 2017.

A. Simplification of Step 2 in Main Paper Section 3

Under the assumption that $\epsilon = 0$, the formula for W_1 in terms of $\mathbb{1}_{\mathcal{N}_k}$ simplifies. We drop the subscript k from $\mathcal{N}$ for readability. When $\epsilon = 0$, there are exactly k samples in the neighborhood set $\mathcal{N}$. Specifically, $\sum_p \mathbb{1}_{\mathcal{N}}(i,p) = k$ and $\sum_p 1 - \mathbb{1}_{\mathcal{N}}(i,p) = n - k$. Eq. 3 becomes:

$$\begin{aligned} W_1(i,j) &= \frac{\mathbb{1}_{\mathcal{N}}(i,j)}{2} \cdot \left(\frac{M_+(i,j)}{k} + \frac{M_-(i,j)}{n-k} \right), \\ \text{where } M_+(i,j) &= \sum_{p=1}^{n} \mathbb{1}_{\mathcal{N}}(i,p)\mathbb{1}_{\mathcal{N}}(j,p), \\ \text{and } M_-(i,j) &= \sum_{p=1}^{n} \left(1 - \mathbb{1}_{\mathcal{N}}(i,p)\right)\left(1 - \mathbb{1}_{\mathcal{N}}(j,p)\right). \end{aligned} \tag{11}$$

Under our assumptions, $M_-(i,j)$ simplifies:

$$\begin{aligned} M_-(i,j) &= \sum_{p=1}^{n} \left(1 - \mathbb{1}_{\mathcal{N}}(i,p) - \mathbb{1}_{\mathcal{N}}(j,p) + \mathbb{1}_{\mathcal{N}}(i,p)\mathbb{1}_{\mathcal{N}}(j,p)\right) \\ &= n - k - k + \sum_{p=1}^{n} \mathbb{1}_{\mathcal{N}}(i,p)\mathbb{1}_{\mathcal{N}}(j,p) \\ &= n - 2k + \langle \mathbb{1}_{\mathcal{N}}(i), \mathbb{1}_{\mathcal{N}}(j) \rangle \end{aligned} \tag{12}$$

Note that $M_+(i,j) = \langle \mathbb{1}_{\mathcal{N}}(i), \mathbb{1}_{\mathcal{N}}(j) \rangle$, by definition. Eq. 11 becomes:

$$W_1(i,j) = \frac{\mathbb{1}_{\mathcal{N}}(i,j)}{2} \cdot \left(\frac{\langle \mathbb{1}_{\mathcal{N}}(i), \mathbb{1}_{\mathcal{N}}(j) \rangle}{k} + \frac{n - 2k + \langle \mathbb{1}_{\mathcal{N}}(i), \mathbb{1}_{\mathcal{N}}(j) \rangle}{n-k} \right) \tag{13}$$

This can be written as Eq. 8 in the main paper, restated here:

$$W_1(i,j) = \mathbb{1}_{\mathcal{N}}(i,j) \left(a \langle \mathbb{1}_{\mathcal{N}}(i), \mathbb{1}_{\mathcal{N}}(j) \rangle + b \right), \text{ where } a = \frac{1}{2k} + \frac{1}{2(n-k)} \text{ and } b = \frac{n-2k}{2(n-k)} \tag{14}$$

Under the assumption that $n > 2k$, $a > 0$ and $b > 0$.

B. Proof of Proposition 4.1

Forward direction If all samples are correctly ranked w.r.t. every other sample within the batch, show that $\mathcal{L}_{\text{context}} = 0$.

Proof Recall that the Proposition assumes that there are exactly k samples from each class in the batch. This means that the k neighborhood set $\mathcal{N}(i)$ of every sample i contains exactly the k samples in the batch with the same label. In other words, $\mathbb{1}_{\mathcal{N}}(i,j) = y_{ij}$, $\forall i,j$. Further, if $\mathbb{1}_{\mathcal{N}}(i,j) = 1$, then the inner product $\langle \mathbb{1}_{\mathcal{N}}(i), \mathbb{1}_{\mathcal{N}}(j) \rangle = k$ because the neighborhood sets of i and j are identical and there are exactly k samples in this set.

When $\langle \mathbb{1}_{\mathcal{N}}(i), \mathbb{1}_{\mathcal{N}}(j) \rangle = k$, $W_1(i,j) = \mathbb{1}_{\mathcal{N}}(i,j)$. When $\mathbb{1}_{\mathcal{N}}(i,j) = 0$, $W_1(i,j) = 0$. Therefore, $W_1(i,j) = \mathbb{1}_{\mathcal{N}}(i,j) = y_{ij}$, $\forall i,j$.

Following Eq. 4, it is also clear that $w_{ij} = W_1(i,j)$ when samples are correctly ranked. This is because, the $\mathcal{R}_{k/2}$ neighborhood is a subset of the larger $\mathcal{N}_k$ neighborhood. That is, if $\mathbb{1}_{\mathcal{R}_{k/2}}(i,j) = 1$, then $\mathbb{1}_{\mathcal{N}}(i,j) = 1$. Since $\mathbb{1}_{\mathcal{N}}(i,p) = \mathbb{1}_{\mathcal{N}}(j,p)$, $\forall p$, when $\mathbb{1}_{\mathcal{N}}(i,j) = 1$, averaging the entries of W_1 over $\mathcal{R}_{k/2}$ as indicated in Eq. 4 does not change W_1. Therefore, $W_2(i,j) = W_1(i,j)$. Finally, y_{ij} is symmetric, so $W_2(i,j) = w_{ij}$. This proves that $y_{ij} = w_{ij}$ and $\mathcal{L}_{\text{context}} = 0$ by Eq. 5.

Backward direction If $\mathcal{L}_{\text{context}} = 0$, show that all samples are correctly ranked w.r.t. every other samples within the batch.

Proof We prove this by contradiction. Suppose that $\mathcal{L}_{\text{context}} = 0$ and there exists a pair of samples i, j where j is not correctly ranked w.r.t. i. There are two possible cases: (1) $y_{ij} = 1$ but $\mathbb{1}_{\mathcal{N}}(i,j) = 0$ or (2) $y_{ij} = 0$ but $\mathbb{1}_{\mathcal{N}}(i,j) = 1$. (1)

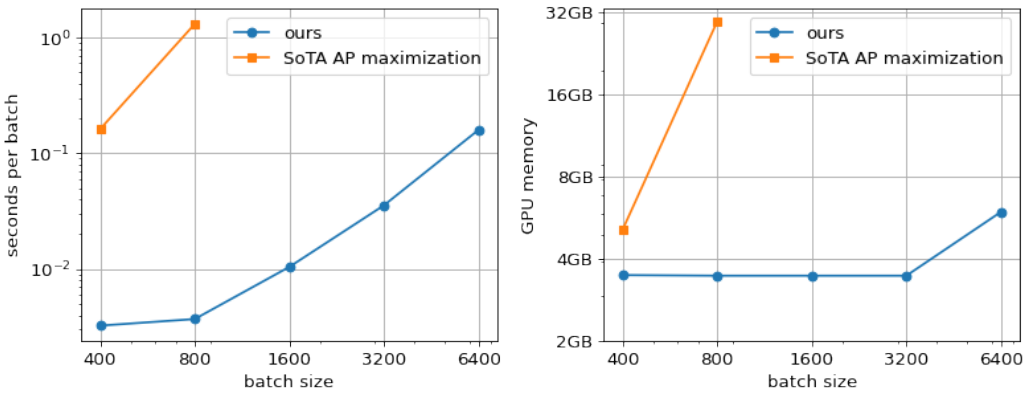

Figure 8. Comparison of scalability in time and space between our loss function and Roadmap. Our method is plotted in blue, while Roadmap is plotted in orange. For this experiment, we pre-calculate the 2048-dimensional ResNet-50 features and train a linear embedding layer using each loss function. Both the time taken per batch (left) and GPU memory used (right) scale cubicly. However, the time and space taken to calculate our loss function is negligible compared to backbone evaluation, even for batch size 6400. On the other hand, the time and space needed to calculate the Roadmap loss quickly becomes unpractical for large batch sizes. Note: we pre-compute all features and store them in GPU memory. The memory used by the pre-computation is freed, but the GPU memory being used, as indicated by the Nvidia tool, remains at the maximum memory used over the lifetime of the process. Therefore, the memory consumed by calculating the loss function is not visible until it exceeds the memory used by the pre-computation.

is the case where i and j have the same label, but j is not ranked as one of the top-k neighbors of i. (2) is the case where i and j have different labels, but j is ranked as one of the top-k neighbors of i. Since we assume that there are exactly k samples with the same label per batch, both (1) and (2) must be true. In particular, assume that (2) is true. Under Eq. 14, if $\mathbb{1}_{\mathcal{N}}(i,j) = 1$, then $W_1(i,j) > 0$ because $\mathbb{1}_{\mathcal{N}}(i,j)$ is multiplied by a non-zero positive number. Similarly, under Eq. 4, if $W_1(i,j) > 0$, then $w_{ij} > 0$ because $W_1(i,j)$ is multiplied by non-zero positive numbers and added to positive numbers. Since $w_{ij} > 0$ and $y_{ij} = 0$, $\mathcal{L}_{\text{context}} > 0$, which contradicts the assumption that $\mathcal{L}_{\text{context}} = 0$ and concludes the proof.

C. Scalability

While the calculation of our loss function scales cubicly in time and space, GPU implementations of matrix multiplication are highly optimized. Consequently, our contextual loss remains practical even for batch sizes as high as 6400. See Figure 8. In comparison, AP surrogates such as Roadmap become impractical beyond batch sizes of about 2000. We used one A40 GPU for this experiment, so we were unable to plot points above 32 GB of GPU memory.

D. Decomposability Gap

Ramzi et al. (2021) demonstrated mathematically the idea of a decomposability gap in their Roadmap paper. When optimizing a ranking metric such as AP, the average batch-wise AP is a loose upper bound of AP over the entire dataset. Given a reasonable embedding space (e.g. halfway through training), most randomly sampled batches are perfectly ranked, even if ranking over the entire dataset is far from optimal. Ramzi et al. (2021) call this difference between the batch-wise objective and the dataset-wise objective the "decomposability gap" and propose to use the standard contrastive loss to reduce this gap. The resulting optimization objective is a convex combination of the ranking loss and $\mathcal{L}_{\text{contrast}}$.

This decomposability gap could partially explain why we need to regularize $\mathcal{L}_{\text{context}}$ with $\mathcal{L}_{\text{contrast}}$. Specifically, $\mathcal{L}_{\text{context}}$ reaches a minimum of zero when all samples within a mini-batch are correctly ranked. At this point, the embedding no longer changes until an incorrectly-ranked mini-batch is sampled. This is problematic, because, without hard off-line sampling, the probability that a randomly sampled batch contains the "hard" triplet that is incorrectly ranked diminishes. This justifies the need for $\mathcal{L}_{\text{contrast}}$ to complement our contextual loss. See Figure 9 for an empirical verification of the existence of a decomposability gap on the CUB dataset.

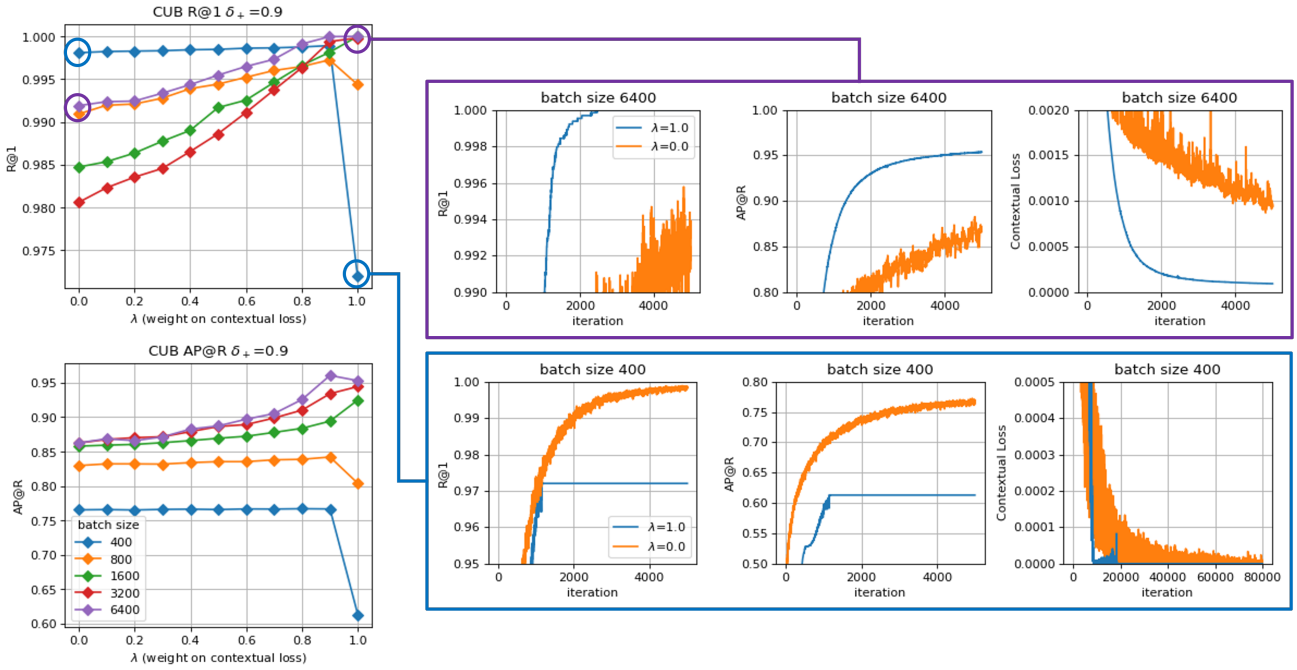

Figure 9. Empirical verification of the existence of a decomposability gap when optimizing the contextual loss. We run this experiment on CUB. We train the linear embedding layer on top of fixed ResNet-50 features. We experiment with different λ. $\lambda = 0$ corresponds to the contrastive loss, while $\lambda = 1$ corresponds to the contextual loss. We experiment with different batch sizes. We only plot the train R@1 accuracy in this figure. Note that batch size of 6400 corresponds to the entire dataset (i.e. decomposability gap is zero). The plots on the right show the R@1, AP@R, and contextual loss over the course of training for $\lambda = 0$ and $\lambda = 1$ for both the largest batch size and the smallest batch size. Observe that when the batch size is 400, optimizing the contextual loss ($\lambda = 1$) in blue does not converge to the correct ranking on the entire dataset. The training stalls at around 1000 iterations. This is because the embedding space after iteration 1000 is good enough in the sense that the batch-wise ranking is perfect for all sampled batches. Meanwhile, if we use batch size 6400 (no mini-batching), the decomposability gap is zero, and the contextual loss in blue clearly converges to the correct ranking over the entire dataset. Further observe that on the larger batch sizes, the optimal train R@1 is achieved when $\lambda = 1$. This suggests that the contrastive loss is not needed on large batch sizes, where the decomposability gap is smaller.

E. Contextual Similarity Definition in Set Notation

In this section, we state the same contextual similarity definition as Section 3 in the main paper, but following the notation of Kim et al. (2022).

We compute the contextual similarity on a single batch, not the entire dataset. Two samples are contextually similar if they share the same neighbors, i.e. the intersection of their k-neighbor sets is large. Following this intuition, we calculate $\tilde{w}_{ij}$, the preliminary contextual similarity between samples i and j:

$$\mathcal{N}_{k+\epsilon}(i) = \{j \mid s_{ij} \leq s_{ip} + \epsilon \text{ where } p \text{ denotes the } k\text{-th closest neighbor of } i\} \tag{15}$$

$$\tilde{w}_{ij} = \begin{cases} |\mathcal{N}_{k+\epsilon}(i) \cap \mathcal{N}_{k+\epsilon}(j)| \;/\; |\mathcal{N}_{k+\epsilon}(i)| & \text{, if } j \in \mathcal{N}_{k+\epsilon}(i) \\ 0 & \text{, otherwise.} \end{cases} \tag{16}$$

We include i in $\mathcal{N}_{k+\epsilon}(i)$ (so if $k = 2$, then $\mathcal{N}_{k+0}(i)$ includes two elements: i and its closest neighbor). Then, we use query expansion and symmeterize to obtain the final contextual similarity. Query expansion (Arandjelović & Zisserman, 2012) is a standard evaluation-time trick in metric learning. It boosts retrieval performance by retrieving neighbors of a sample's neighbors. Analogously, we adjust the contextual similarity by averaging $\tilde{w}_{ij}$ over the set of close reciprocal neighbors $\mathcal{R}_{k/2}(i)$.

$$\mathcal{N}_{k/2+\epsilon}(i) = \{j \mid s_{ij} \leq s_{ip} + \epsilon \text{ where } p \text{ denotes the } k/2\text{-th closest neighbor of } i\} \tag{17}$$

$$\mathcal{R}_{k/2+\epsilon}(i) = \{j \mid j \in \mathcal{N}_{k/2+\epsilon}(i) \text{ and } i \in \mathcal{N}_{k/2+\epsilon}(j)\} \tag{18}$$

$$\hat{w}_{ij} = \frac{1}{|\mathcal{R}_{k/2+\epsilon}(i)|} \sum_{p \in \mathcal{R}_{k/2+\epsilon}(i)} \tilde{w}_{pj} \quad , \quad w_{ij} = \frac{1}{2}(\hat{w}_{ij} + \hat{w}_{ji}). \tag{19}$$

Note that $w_{ij} \in [0, 1]$ and only depend on the cosine similarities s_{ij} between embeddings. We want to optimize the embeddings such that w_{ij} converges to y_{ij}. The value of k is not arbitrary; it must be set to the number of samples per label in the mini-batch. For example, a standard metric learning setup is to randomly sample 32 labels and then sample 4 images per label. In this case, we set $k = 4$.

E.1. Differences with STML

This subsection enumerates in detail the technical differences between our definition of contextual similarity (w_{ij} in Eq. 19) and the definition of contextual similarity in STML (Kim et al., 2022). Note that by necessity, we reference equations and notation in the STML paper.

1. STML averages a non-linear function of the cosine similarity (w_{ij}^P in their Eq. 5) with the contextual similarity (w_{ij}^C in their Eq. 5) to arrive at an estimate of the ground-truth similarity. In their work, the resulting similarity is used to "teach" a student embedding network. In our work, the contextual similarity is optimized directly by the contextual loss $\mathcal{L}_{\text{context}}$, and the cosine similarity is directly optimized by the contrastive loss $\mathcal{L}_{\text{contrast}}$.
2. We add a margin term ϵ to the neighborhood definition; STML uses $\epsilon = 0$. In our loss, the margin is necessary to encourage separation between embeddings with different labels.
3. (Minor) We use the set of $k + \epsilon$ neighbors to calculate $\tilde{w}_{ij}$ in our Eq. 16; STML uses the set of k *reciprocal* neighbors to calculate $\tilde{w}_{ij}$ in their Eq. 7.
4. (Minor) We use the set of k *reciprocal* neighbors for query expansion in our Eq. 18; STML uses the $k/2$ neighbors in their Eq. 8.

Additionally, our work contains the following technical innovations not found in (Kim et al., 2022):

- The implementation of contextual similarity in (Kim et al., 2022) involves indexing a matrix and setting those values to a constant. Consequently, the output depends on the input *indirectly* through an indexing operation. We re-implement the contextual similarity calculation in a way that only uses greater-than, logical-and, addition, and multiplication. A detailed explanation of this process is detailed in Appendix E.2 and E.3.

- We determine suitable differentiable substitutions for the logical-and and greater-than functions. logical-and could be replaced by simple averaging, multiplication, or $\min$. greater-than can be replaced by a sigmoid, a smooth upper-bound, or the constant-gradient heaviside. We ultimately chose to use multiplication and a constant-gradient heaviside. These two choices are intuitively motivated in Appendix E.2, Fig. 10, and experimentally justified in Table 5.
- (Minor) Optimizing the intersection of neighborhood sets in Eq. 16 directly is problematic and leads to shrinkage of the embedding space. We mitigate this issue by additionally optimizing the intersection of the complements of the neighborhood sets. This is explained in Figure 12

E.2. Detailed Optimization Motivation

The definition in the previous sub-section clearly contains three discrete operations: (1) greater-than, (2) logical-and, and (3) intersection. For optimization, we will deal extensively with indicator matrices: we denote as $\mathbb{1}_{\mathcal{N}} \in \mathbb{R}^{n \times n}$ the indicator matrix where $\mathbb{1}_{\mathcal{N}}(i,j) = 1$ if $j \in \mathcal{N}(i)$ for some set $\mathcal{N}$. We use $\odot$ to denote element-wise multiplication.

Greater-than This is used in Eq. 15 and 17 and is equivalent to the non-differentiable heaviside function θ. Our approach is to use the exact value of $\theta(\cdot)$ in the forward pass and a constant positive gradient α in the backward pass. A constant positive gradient is reasonable since θ is a (non-strictly) increasing function.

$$\text{Forward: } \theta(x) = 1 \text{ if } x \geq 0 \,;\, 0 \text{ otherwise} \quad \text{Backward: } \frac{\partial \theta(x)}{\partial x} = \alpha. \tag{20}$$

In Section F.1 we show with a toy experiment that this approach is robust despite being somewhat heuristic. In contrast, θ is traditionally approximated by a sigmoid (e.g. for AP approximation and in Gated Recurrent Networks (Cho et al., 2014)):

$$\theta_\sigma(x) = \frac{1}{1 + \exp\left(x/\tau\right)}. \tag{21}$$

θ_σ trades off the quality of the approximation with the domain where gradients are non-zero. As temperature τ decreases, θ_σ approaches θ, but gradient vanishes everywhere except in a small region around the boundary. This behavior is not intuitive: it is undesirable to only have a large gradient at the boundary. Some prior work (e.g. ROADMAP) side-step this issue by using an upper-bound to the heaviside function (where the right side of the heaviside function increases linearly), which solves the gradient issue at the expense of grossly over-estimating the true objective. In our case, this is especially concerning since we will be multiplying together indicator functions. In Section G.1, we show that our approach in Eq. 20 achieves better empirical results than a sigmoid approximation.

Logical-and A logical-and is used explicitly in Eq. 18 and implicitly in Eq. 16 and 19. There are two differentiable substitutes for logical-and: $\min$ and multiplication. Multiplication is smooth, but has no gradient at the origin. $\min$ is logically consistent on the continuous domain [0,1], but the gradient is not continuous when inputs are equal. We found experimentally that multiplication is the best option, see row 5 of Table 5. A gradient of zero at the origin is desireable in the case of Eq. 18 and 19 (query expansion step). A work-around for the zero-gradient issue for Eq. 16 is presented in the next sub-section.

Intersection The number of elements in the intersection in Eq. 16 is calculated using matrix multiplication. A sample $p \in \mathcal{N}_{k+\epsilon}(i) \cap \mathcal{N}_{k+\epsilon}(j)$ if $p \in \mathcal{N}_{k+\epsilon}(i)$ and $p \in \mathcal{N}_{k+\epsilon}(j)$. Using multiplication for the logical-and, the number of elements in the intersection can be expressed compactly as a function of the indicator matrices:

$$M_+ = \mathbb{1}_{\mathcal{N}_{k+\epsilon}} \mathbb{1}^\intercal_{\mathcal{N}_{k+\epsilon}}, \text{ where } M_+(i,j) = \left|\mathcal{N}_{k+\epsilon}(i) \cap \mathcal{N}_{k+\epsilon}(j)\right|. \tag{22}$$

We are now ready to state the loss function by combining the definition of contextual similarity with the optimization method.

E.3. Loss in Matrix Notation

Our loss function straightforwardly combines the previous two sub-sections, with one exception for the intersection calculation in Eq. 16. Optimizing the intersection with M_+ tends to focus on pairs of neighboring samples because using multiplication for logical-and zeros out the gradient when both inputs are 0. This is problematic since the resulting embedding becomes clustered regardless of true similarity (see Figure 12). We mitigate this problem by optimizing the

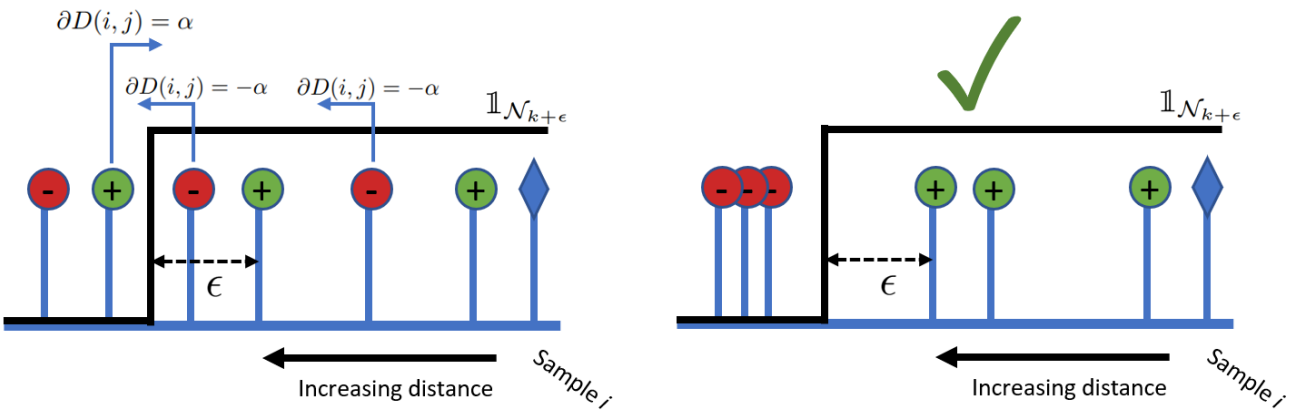

Figure 10. Illustration of $\mathcal{N}_{k+\epsilon}$ optimization using the simplified loss function $\mathcal{L}_1$ in Eq. 32. Negative samples which are in the set $\mathcal{N}_{k+\epsilon}(i)$ are pushed away from i by a constant gradient. Positive samples which are not in the set $\mathcal{N}_{k+\epsilon}(i)$ are pulled closer. This behavior is analogous to the standard contrastive loss: $\mathcal{L}_{\text{contrast}}$ applies a constant gradient to pairs violating a fixed margin, while $\mathcal{L}_1$ applies a constant gradient to pairs violating the flexible margin defined by the k-th neighbor. $k = 4$ in the illustration.

intersection of the complements $\mathcal{N}^c_{k+\epsilon}(i) \cap \mathcal{N}^c_{k+\epsilon}(j)$ in addition to optimizing the original intersection in Eq. 16. ^{c} denotes the complement of a set. Maximizing the size of the intersection between two sets is equivalent to maximizing the size of the intersection between their complements, and vice versa. This can be trivially proven under the assumptions that the universal set is constant and that the size of both sets is constant.

Let $D \in \mathcal{R}^{n \times n}$ denote the matrix where $D(i,j) \in [0,4]$ is the squared Euclidean distance between the normalized features of sample i and j. $D(i,j) = 2 - 2s_{ij}$. sg denotes stop gradient.

Step 1 (Neighborhood Optimization):

$$\mathbb{1}_{\mathcal{N}_{k+\epsilon}}(i,j) = \theta(-D(i,j) + \text{sg}(D(i,p)) + \epsilon) \text{ where } p \text{ denotes the } k\text{-th closest neighbor of } i. \tag{23}$$

Step 2 (Optimizing Intersection of Neighborhoods):

$$\tilde{W} = \frac{1}{2}\left(\frac{M_+}{\text{sg}(|\mathcal{N}_{k+\epsilon}(i)|)} + \frac{M_-}{\text{sg}(|\mathcal{N}^c_{k+\epsilon}(i)|)}\right) \odot \mathbb{1}_{\mathcal{N}_{k+\epsilon}} \quad \text{where } M_+ = \mathbb{1}_{\mathcal{N}_{k+\epsilon}} \mathbb{1}^\intercal_{\mathcal{N}_{k+\epsilon}} \text{ and } M_- = \mathbb{1}_{\mathcal{N}^c_{k+\epsilon}} \mathbb{1}^\intercal_{\mathcal{N}^c_{k+\epsilon}}. \tag{24}$$

Step 3 (Query Expansion):

$$\mathbb{1}_{\mathcal{N}_{k/2+\epsilon}}(i,j) = \theta(-D(i,j) + \text{sg}(D(i,p)) + \epsilon) \text{ where } p \text{ denotes the } k/2\text{-th closest neighbor of } i \tag{25}$$

$$\mathbb{1}_{\mathcal{R}_{k/2+\epsilon}} = \mathbb{1}_{\mathcal{N}_{k/2+\epsilon}} \odot \mathbb{1}^\intercal_{\mathcal{N}_{k/2+\epsilon}} \tag{26}$$

$$\hat{W} = \frac{\mathbb{1}_{\mathcal{R}_{k/2+\epsilon}} \tilde{W}}{|\mathcal{R}_{k/2+\epsilon}(i)|}, \; W = \frac{1}{2}\left(\hat{W} + \hat{W}^\intercal\right). \tag{27}$$

Finally, we use the MSE loss to optimize $w_{ij} := W(i,j)$ against the true similarity labels y_{ij}:

$$\mathcal{L}_{\text{context}} = \frac{1}{n^2} \sum_{i,j|i \neq j} (y_{ij} - w_{ij})^2. \tag{28}$$

$\mathcal{L}_{\text{context}}$ is the key to our framework and works reasonably on its own. However, it has two vulnerabilities: (1) similar to learning to rank losses, $\mathcal{L}_{\text{context}}$ can be small even if the distance between positive pairs is large, so long as there are no closer negative pairs (see Figure 10); (2) $\mathcal{L}_{\text{context}}$ tends to converge to a solution where the average cosine similarity between all pairs is large, suggesting that only a small portion of the available embedding space is utilized. In response to problem (1),

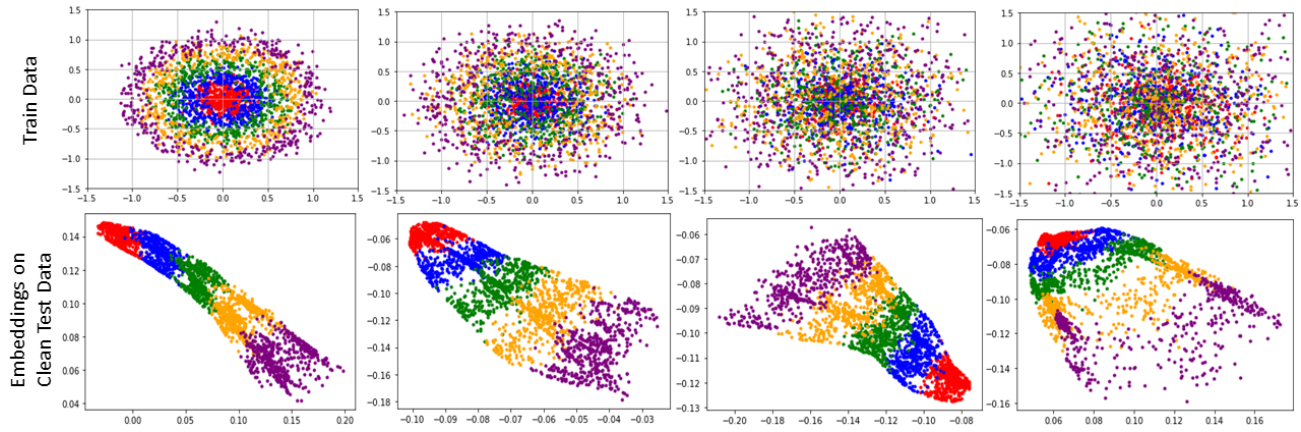

Figure 11. This figure justifies the non-standard approach of optimizing $\mathcal{L}_1$ (Eq. 32) by overriding the gradient of θ. We generate synthetic 2-D data consisting of five concentric circles; each color represents a label. We use a standard three layer MLP with ReLU activations to output a 2D embedding. We train on increasingly noisy data and plot the resulting embeddings on clean data. We only sample 4 points per label per batch ($k = 4$). $\mathcal{L}_1$ minimization results in reasonable embeddings despite its simplicity.

we add the standard contrastive loss ((Hadsell et al., 2006)), which explicitly optimizes the cosine similarity toward fixed margins δ_+ and δ_-. In response to problem (2), we add a non-standard but straightforward similarity regularizer, which regularizes the average cosine similarity between all pairs toward a fixed value $\tilde{s}$. Our final framework (Eq. 31) minimizes a weighted combination of the three losses.

$$\mathcal{L}_{\text{contrast}} = \frac{\sum_{i,j|y_{ij}=1}(\delta_+ - s_{ij})_+}{\left|\{i,j|y_{ij}=1 \text{ and } \delta_+ - s_{ij} > 0\}\right|} + \frac{\sum_{i,j|y_{ij}=0}(s_{ij} - \delta_-)_+}{\left|\{i,j|y_{ij}=0 \text{ and } s_{ij} - \delta_- > 0\}\right|} \tag{29}$$

$$\mathcal{L}_{\text{reg}} = \left(\tilde{s} - \frac{1}{n^2}\sum_{i,j}^{n^2} s_{ij}\right)^2 \tag{30}$$

$$\mathcal{L}_{\text{ours}} = \lambda\mathcal{L}_{\text{context}} + (1-\lambda)\mathcal{L}_{\text{contrast}} + \gamma\mathcal{L}_{\text{reg}}. \tag{31}$$

Note that El-Nouby et al. (2021) propose a superficially similar similarity regularization scheme, which "maximizes the distance between every point and its nearest neighbor". Our $\mathcal{L}_{\text{reg}}$ regularizes the average similarity between all pairs of samples. Nearest neighbor pairs are dominated by positive pairs, while all pairs are dominated by negative pairs. Therefore, our regularizer is fundamentally different from El-Nouby et al. (2021).

F. Slightly Different Analysis

The contextual similarity loss function $\mathcal{L}_{\text{context}}$ is highly non-trivial. In this section we demystify each of the three steps using a toy experiment and gradient analysis. This is a slightly different analysis than the one given in the main paper.

F.1. Step 1: Neighborhood optimization

Consider the following simplified version of $\mathcal{L}_{\text{context}}$:

$$\mathcal{L}_1 = \mathcal{L}_{\text{MSE}}(y_{ij}, \mathbb{1}_{\mathcal{N}_{k+\epsilon}}(i,j)). \tag{32}$$

This is a valid loss function that works reasonably on its own (see Figure 11). During each iteration, we sample a batch with exactly k samples from each label. Intuitively, $\mathcal{L}_1$ reaches a minimum of zero when $\mathcal{N}_{k+\epsilon}(i)$ includes only i and the $k-1$ other samples with the same label, i.e. all samples are correctly ranked. Using the gradient of $\theta(\cdot)$ defined in Eq. 20, we see that negative samples which are in $\mathcal{N}_{k+\epsilon}(i)$ receive a gradient of magnitude α pushing it away from i, while positive samples which are not in $\mathcal{N}_{k+\epsilon}(i)$ receive a gradient of magnitude α pushing it toward i. See Figure 10.

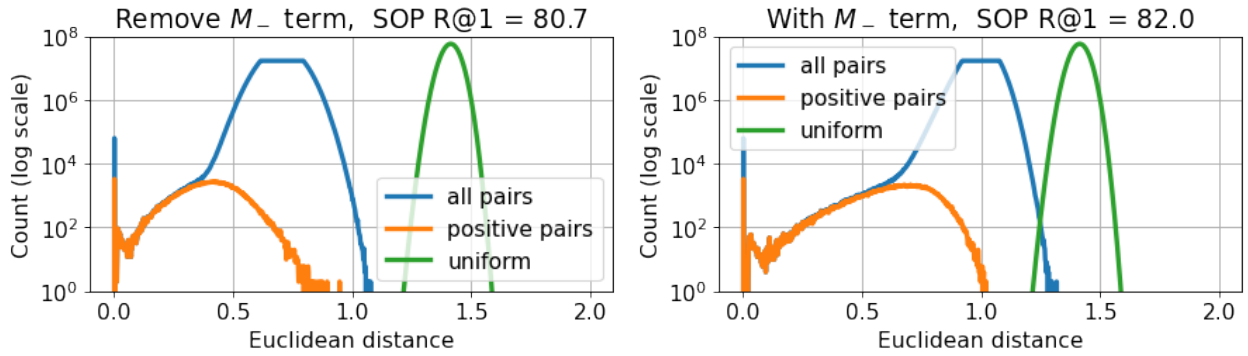

Figure 12. This figure justifies the M_- term in Eq. 24. The left plot shows the distribution of distances between pairs in normalized embedding space when we only optimize M_+; the right plot shows the same when we optimize both M_- and M_+. A uniformly distributed embedding space has an average pair-wise distance of $\sqrt{2}$ because most directions are orthogonal in high dimensions. This is indicated by the green distribution. Clearly, more of the embedding space is utilized when we optimize both M_- and M_+.

F.2. Step 2: Optimizing Intersection of neighborhoods

While $\lambda\mathcal{L}_1 + (1-\lambda)\mathcal{L}_{\text{contrastive}}$ is a reasonable loss function already, row 1 of Table 5 shows that $\mathcal{L}_1$ collapses without the contrastive loss ($\lambda = 1$). We propose that this is because $\mathcal{L}_1$ provides very sparse gradients: $\partial\mathcal{L}_1/\partial s_{ij} = 0$ for most (i,j) pairs. Consider the following more complicated loss:

$$\mathcal{L}_2 = \mathcal{L}_{\text{MSE}}(y_{ij}, \tilde{W}(i,j)). \tag{33}$$

$\tilde{W}$ is defined in Eq. 24. $\tilde{W}(i,j)$ has two terms multiplied together element-wise. The $\mathbb{1}_{\mathcal{N}_{k+\epsilon}}$ term was already addressed in the previous sub-section, so we focus on the first term, which optimizes the intersection between neighborhood sets M_+. We offer a straight-forward intuition: *Maximizing the intersection between the neighborhood sets of two samples is equivalent to pushing one sample towards the context of the other sample and vice versa*; *minimizing the intersection is equivalent to pulling apart the contexts of the two samples*. We show this intuition by analyzing the gradient.

For simplicity, consider $\epsilon = 0$, such that the normalization factors in Eq. 24 are equal to k and $n-k$, resp. Further consider a negative pair of samples i and j ($y_{ij} = 0$), where j is wrongly ranked w.r.t. i (i.e. $j \in \mathcal{N}_k(i)$). Following the loss function equations:

$$\begin{aligned} M_+(i,j) &= \langle \mathbb{1}_{\mathcal{N}_k}(i), \mathbb{1}_{\mathcal{N}_k}(j) \rangle \qquad M_-(i,j) = \langle \mathbb{1}^c_{\mathcal{N}_k}(i), \mathbb{1}^c_{\mathcal{N}_k}(j) \rangle \\ \tilde{W}(i,j) &= \frac{1}{2}\left(\frac{1}{k}M_+(i,j) + \frac{1}{n-k}M_-(i,j)\right) \cdot \underbrace{\mathbb{1}_{\mathcal{N}_k}(i,j)}_{=1}. \end{aligned} \tag{34}$$

$\tilde{W}(i,j) \in (0,1]$ must be a non-zero positive number. Using the equation for $\mathcal{L}_2$, we see that the gradient w.r.t. $\tilde{W}(i,j)$ must be positive because $y_{ij} = 0$. We are now ready for the backward pass. For simplicity of notation, $\partial\tilde{W}(i,j) := g_{ij} > 0$ denotes the gradient of the loss w.r.t. $\tilde{W}(i,j)$.

$$\begin{aligned} \partial M_+(i,j) &= \frac{1}{2k}g_{ij}, \text{ and } \partial M_-(i,j) = \frac{1}{2(n-k)}g_{ij} \\ \partial\mathbb{1}_{\mathcal{N}_k}(i) &= \frac{1}{2k}g_{ij}\mathbb{1}_{\mathcal{N}_k}(j)\,,\ \partial\mathbb{1}_{\mathcal{N}_k}(j) = \frac{1}{2k}g_{ij}\mathbb{1}_{\mathcal{N}_k}(i) \\ \partial\mathbb{1}^c_{\mathcal{N}_k}(i) &= \frac{1}{2(n-k)}g_{ij}\mathbb{1}^c_{\mathcal{N}_k}(j)\,,\ \partial\mathbb{1}^c_{\mathcal{N}_k}(j) = \frac{1}{2(n-k)}g_{ij}\mathbb{1}^c_{\mathcal{N}_k}(i). \end{aligned} \tag{35}$$

Use the fact that $\mathbb{1}_{\mathcal{N}^c_k} = 1 - \mathbb{1}_{\mathcal{N}_k}$ and $\partial\mathbb{1}_{\mathcal{N}_k} = -\partial\mathbb{1}_{\mathcal{N}^c_k}$ and vice versa:

$$\begin{cases} \partial\mathbb{1}_{\mathcal{N}_k}(i) & = \frac{g_{ij}}{2k}\left(\mathbb{1}_{\mathcal{N}_k}(j)\right) \\ \partial\mathbb{1}_{\mathcal{N}_k}(j) & = \frac{g_{ij}}{2k}\left(\mathbb{1}_{\mathcal{N}_k}(i)\right) \end{cases} \tag{36}$$

Table 4. R@k Results. We use ResNet-50 with an embedding size of 512 for all experiments. † indicates results reported by the original authors; we re-run all other baselines using the implementation by (Musgrave et al., 2020b). Standard deviations are based on three trials with the same train-test split. We emphasize that our R@1 performance is at least two standard deviations better than the next best baseline on all datasets. We are at least comparable to baselines on other R@k metrics.

	CUB				Cars			
Method	R@1	R@2	R@4	R@8	R@1	R@2	R@4	R@8
Contrastive	68.5 ± 0.3	78.3 ± 0.1	86.0 ± 0.2	91.3 ± 0.1	85.4 ± 0.2	91.1 ± 0.3	94.6 ± 0.3	96.8 ± 0.1
Triplet	67.3 ± 0.2	77.9 ± 0.1	85.6 ± 0.2	91.2 ± 0.1	77.6 ± 1.3	85.4 ± 0.8	90.8 ± 0.7	94.1 ± 0.4
NtXent	65.7 ± 0.4	76.3 ± 0.2	84.3 ± 0.4	90.0 ± 0.4	79.0 ± 0.6	86.0 ± 0.3	91.0 ± 0.2	94.4 ± 0.3
MS	68.9 ± 0.5	78.5 ± 0.4	86.0 ± 0.6	91.4 ± 0.5	88.7 ± 0.4	93.0 ± 0.2	95.7 ± 0.1	97.3 ± 0.1
N-Softmax†	61.3	73.9	83.5	90.0	84.2	90.4	94.4	96.9
Proxy NCA ++†	69.0 ± 0.8	**79.8** ± 0.7	**87.3** ± 0.7	**92.7** ± 0.4	86.5 ± 0.4	92.5 ± 0.3	95.7 ± 0.2	**97.7** ± 0.1
Fast-AP	63.3 ± 0.1	73.7 ± 0.4	82.2 ± 0.3	88.5 ± 0.2	74.7 ± 0.4	82.5 ± 0.7	88.0 ± 0.6	92.2 ± 0.2
Smooth-AP	66.5 ± 0.9	76.6 ± 0.5	84.8 ± 0.6	90.8 ± 0.4	81.1 ± 0.2	87.8 ± 0.4	92.2 ± 0.3	95.1 ± 0.3
ROADMAP	68.7 ± 0.5	78.3 ± 0.3	86.1 ± 0.3	91.1 ± 0.1	84.5 ± 0.5	90.3 ± 0.0	93.9 ± 0.0	96.2 ± 0.1
Ours	**69.8** ± 0.2	**79.8** ± 0.1	**87.1** ± 0.1	92.3 ± 0.2	**89.3** ± 0.0	**93.7** ± 0.2	**96.3** ± 0.1	**97.8** ± 0.2

	SOP			mini-iNaturalist			
Method	R@1	R@10	R@100	R@1	R@4	R@16	R@32
Contrastive	82.4 ± 0.0	91.9 ± 0.0	96.0 ± 0.0	43.5 ± 0.1	62.7 ± 0.1	77.6 ± 0.1	83.2 ± 0.1
Triplet	82.0 ± 0.0	92.5 ± 0.1	96.7 ± 0.0	35.4 ± 0.1	56.5 ± 0.1	74.7 ± 0.1	81.7 ± 0.1
NtXent	79.7 ± 0.2	90.8 ± 0.0	96.1 ± 0.0	40.8 ± 0.1	61.6 ± 0.1	78.0 ± 0.0	83.9 ± 0.0
MS	81.4 ± 0.0	91.4 ± 0.0	96.1 ± 0.1	44.9 ± 0.1	63.9 ± 0.1	78.4 ± 0.1	83.9 ± 0.1
N-Softmax†	78.2	90.6	96.2	–	–	–	–
Proxy NCA ++†	80.7 ± 0.5	92.0 ± 0.3	96.7 ± 0.1	–	–	–	–
Fast-AP	80.3 ± 0.1	91.0 ± 0.1	96.0 ± 0.0	35.6 ± 0.2	55.8 ± 0.1	72.8 ± 0.0	79.3 ± 0.0
Smooth-AP	82.0 ± 0.0	92.6 ± 0.0	**96.9** ± 0.0	42.7 ± 0.0	63.3 ± 0.0	79.0 ± 0.0	84.7 ± 0.0
ROADMAP	83.1 ± 0.1	92.6 ± 0.0	96.6 ± 0.0	45.9 ± 0.1	**65.8** ± 0.0	**80.4** ± 0.1	**85.7** ± 0.0
Ours	**83.3** ± 0.0	**92.9** ± 0.1	96.7 ± 0.0	**46.2** ± 0.0	**65.8** ± 0.1	80.2 ± 0.1	85.4 ± 0.1

$$\begin{cases} \partial \mathbb{1}_{\mathcal{N}_k}(i) & = \frac{g_{ij}}{2(n-k)} \left(\mathbb{1}_{\mathcal{N}_k}(j) - 1 \right) \\ \partial \mathbb{1}_{\mathcal{N}_k}(j) & = \frac{g_{ij}}{2(n-k)} \left(\mathbb{1}_{\mathcal{N}_k}(i) - 1 \right) \end{cases} \tag{37}$$

Eq. 36 is the contribution to the gradient from optimizing $\mathbb{1}_{\mathcal{N}_k}$, while Eq. 37 is the contribution to the gradient from optimizing $\mathbb{1}^c_{\mathcal{N}_k}$. These can be added together:

$$\begin{cases} \partial \mathbb{1}_{\mathcal{N}_k}(i) & = \frac{g_{ij}}{2(n-k)} \left(\mathbb{1}_{\mathcal{N}_k}(j) - 1 \right) + \frac{g_{ij}}{2k} \left(\mathbb{1}_{\mathcal{N}_k}(j) \right) \\ \partial \mathbb{1}_{\mathcal{N}_k}(j) & = \frac{g_{ij}}{2(n-k)} \left(\mathbb{1}_{\mathcal{N}_k}(i) - 1 \right) + \frac{g_{ij}}{2k} \left(\mathbb{1}_{\mathcal{N}_k}(i) \right) \end{cases} \tag{38}$$

Going backward through $\theta(\cdot)$ multiplies the gradient by $-\alpha$ (negative sign accounts for optimizing distance instead of similarity):

$$\partial D(i) = \frac{-\alpha g_{ij}}{2(n-k)} \left(\mathbb{1}_{\mathcal{N}_k}(j) - 1 \right) - \frac{\alpha g_{ij}}{2k} \left(\mathbb{1}_{\mathcal{N}_k}(j) \right) \text{ , and vice versa for } \partial D(j). \tag{39}$$

From the above equation, $\partial D(i,p) < 0$ when $\mathbb{1}_{\mathcal{N}_k}(j,p) = 1$ and $\partial D(i,p) > 0$ when $\mathbb{1}_{\mathcal{N}_k}(j,p) = 0$, for all samples p in the batch. In words, we increase the distance between i and all samples in the neighborhood of j, and we decrease the distance between i and all points outside the neighborhood of j. Recall that we assumed i and j to be a negative pair, so it makes sense to pull i away from the context of j in this fashion.

F.3. Step 3: Query Expansion

Query expansion (QE) is an established trick for image retrieval (Arandjelović & Zisserman, 2012). QE expands the neighborhood set by additionally retrieving the neighbors of very close neighbors. In the unsupervised setting, QE refers to averaging the contextual similarity scores for samples in the $\mathcal{R}_{k/2+\epsilon}$ neighborhood (see Eq. 27); this leads to more robust pseudo-supervision. In our setting, we have two reasons for using QE (Eq. 25 26 and 27): (1) averaging $\tilde{w}_{ij}$ with very close neighbors could have the same effect as label smoothing and (2) the $\mathcal{N}_{k/2+\epsilon}$ neighborhood is optimized by Eq. 26 and 27.

Table 5. Ablation Results. Here, we experiment with variations of $\mathcal{L}_{\text{context}}$, both by itself ($\lambda = 1$) and regularized by a small amount of contrastive loss ($\lambda = 0.8$). We exhaustively test various ways to simplify or modify the contextual loss presented in Eq. 23 - 28. On the SOP dataset, we show that all of the modifications to $\mathcal{L}_{\text{context}}$ decrease R@1.

Ablation ($\gamma = 0$)		SOP R@1 $\lambda = 0.8$	SOP R@1 $\lambda = 1.0$	Explanation
$\lambda\mathcal{L}_1$	$+(1-\lambda)\mathcal{L}_{\text{contrast}}$	83.13	41.99	Eq. 32 (step 1 only)
$\lambda\mathcal{L}_{1,\sigma}$	$+(1-\lambda)\mathcal{L}_{\text{contrast}}$	79.20	75.61	Eq. 32 but using θ_σ in Eq. 21
$\lambda\mathcal{L}_2$	$+(1-\lambda)\mathcal{L}_{\text{contrast}}$	82.39	81.05	Eq. 33 (skip step 3)
$\tilde{W} = \mathbb{1}_{\mathcal{N}_{k+\epsilon}}$ (skip step 2)		82.74	81.74	
$\lambda\mathcal{L}_{\text{context,min}}$	$+(1-\lambda)\mathcal{L}_{\text{contrast}}$	66.57	–	Use min instead of $\odot$ for logical-and
$\lambda\mathcal{L}_{\text{context},\sigma}$	$+(1-\lambda)\mathcal{L}_{\text{contrast}}$	82.39	77.84	Use θ_σ for all steps
$\lambda\mathcal{L}_{\text{context},M_+}$	$+(1-\lambda)\mathcal{L}_{\text{contrast}}$	82.31	80.72	Remove M_- term from Eq. 24
No stop gradient in Eq. 24		73.03	–	
$\lambda\mathcal{L}_{\text{context}}$	$+(1-\lambda)\mathcal{L}_{\text{contrast}}$	**83.20**	**82.04**	Final results without $\mathcal{L}_{\text{reg}}$

Empirically, QE is necessary to achieve the optimal performance of our framework, since $\mathcal{L}_{\text{context}}$ achieves higher R@1 than $\mathcal{L}_2$ in Table 5.

G. 256 × 256 Resolution Experiments

We include 256 × 256 image resolution results in this section. This setup is slightly different from the main paper. These results are comparable to the results reported in the ROADMAP paper.

Hyperparameters and Setup on 256 × 256 Experiments We use hyperparameter values that are approximately optimal across all datasets. $\lambda = 0.4$, $\gamma = 0.1$, $\alpha = 10.0$, $\epsilon = 0.05$, $k = 4$, $\delta_+ = 0.75$, $\delta_- = 0.6$, $\tilde{s} = 0.25$. We follow the same training procedure as ROADMAP, but with a slightly faster learning-rate schedule for time efficiency. We use Adam with a learning-rate schedule that multiplies the learning rate by 0.3 at Epochs 15, 30, and 45. We train for 80 epochs. We report results on the model with the best test R@1 metric, as is standard in the literature. We use an initial learning rate of 8×10^{-5} on CUB, 0.00016 on Cars, 4×10^{-5} on SOP, and 8×10^{-5} on iNaturalist. We use a batch size of 256 for iNaturalist and 128 for other datasets; the larger batch size is necessary to achieve reasonable performance on iNaturalist. We use a 4 per class sampler. For CUB and Cars, we use random sampling (sample 32 classes at random, then sample 4 images per class). For SOP and iNaturalist, we use hierarchical sampling (Cakir et al. (2019)), following prior work. We use an embedding size of 512 and ResNet-50 with a linear embedding layer. Following prior work, we use layer-norm and max-pooling on the smaller CUB and Cars datasets. We always freeze batch-norm.

Discussion for 256 × 256 Experiments Our R@1 results are at least two standard deviations better than the best baseline across all datasets. Our results are especially good on CUB and Cars, where we achieve R@1 gains of 0.8% and 0.6 %, resp. We achieve more modest R@1 gains of 0.2% and 0.3% on SOP and iNaturalist, resp. We note that these gains are significant because the standard deviation is relatively small. We also note that while Proxy NCA ++ and MS are the best baselines on CUB and Cars, ROADMAP is the best baseline on SOP and iNaturalist. Our method is the best across *all* datasets, suggesting that it is more versatile.

G.1. Ablation Results

In Table 5, we evaluate the contribution of each step in the calculation of $\mathcal{L}_{\text{context}}$. These experiments focus on dissecting $\mathcal{L}_{\text{context}}$, so we set $\gamma = 0$ (no similarity regularizer) and $\lambda = 0.8$ or 1.0. Overall, all of the modifications tested in Table 5 decrease the performance of $\mathcal{L}_{\text{context}}$ in terms of R@1. In particular: row 1 shows that including only step 1 leads to a collapsed representation when $\lambda = 1.0$; rows 3 and 4 show that steps 3 and 2 of the loss calculation are necessary; row 7 shows that the M_- term in Eq. 24 is necessary; rows 2 and 6 show that our approach to optimizing θ in Eq. 20 is better than using a sigmoid.

H. Minor Experimental Details

H.1. Code and Environment

We include code with our submission. We run experiments on 1 V100 GPU with 16 GB of memory. The CUB and Cars experiments take under one hour. The SOP and iNaturalist experiments take 4 hours and 6 hours, respectively. Some of our code is borrowed from ROADMAP. For faster experimentation, we use mixed precision floating point. Experiments take more than 12 hours on P100 GPUs, partially because mixed precision arithmetic does not appear to speed-up experiments as much on P100 GPUs compared to on V100 GPUs.

H.2. Augmentation (Specific to 256 $\times$ 256 Experiments)

On CUB, Cars and SOP, we use random resized crop with default parameters to crop the image to 256 $\times$ 256 pixels. We horizontally flip the image with 50% probability. For testing, we resize the image to 288 pixels, then center crop to 256 pixels. On iNaturalist, we random resize crop to 224 $\times$ 224 pixels, then flip horizontally with 50% probability. We use a smaller image size on iNaturalist so that the larger batch size can fit in the 16 GB of GPU memory. For testing, we resize the image to 256 pixels then center crop to 224 pixels.

H.3. Sampling (Specific to 256 $\times$ 256 Experiments)

We use a batch size of 128 on CUB, Cars and SOP. We use a batch size of 256 on iNaturalist. On all datasets, we train for 80 epochs. An epoch is defined as 15, 30, 655, and 576 batches for CUB, Cars, SOP, and iNaturalist respectively. On CUB and Cars, we use a random 4 per class sampler. On SOP and iNaturalist, we use a balanced hierarchical sampling strategy, since there are super-labels. In particular, half of each batch is randomly sampled from one super-label, and the other half is randomly sampled from another super-label. In order to sample in a balanced manner (to prevent over-emphasis of uncommon super-labels), we arrange all samples in the training dataset into half-batches with same super-labels at the beginning of each epoch. We then form batches by pairing up half-batches in a round-robin fashion.

H.4. Loss Plot

Figure 15 plots the contextual loss and test R@1 over the course of training on SOP, with varying λ. $\gamma = 0$. These plots show that the loss decreases over the course of training; in general, a lower contextual loss corresponds to a higher test R@1. This result suggests that the contextual loss is a reasonable objective for image retrieval.

H.5. Note on Contrastive Loss

The contrastive loss has two margin parameters δ_- and δ_+. The optimal values for δ_+, the positive margin, is different depending on the value of γ (how much similarity regularization is used). For the "contrastive" results in row 1 of Table 4, we use $\delta_- = 0.6$ and $\delta_+ = 0.9$. For our results where $\gamma = 0.1$, we set $\delta_- = 0.6$ and $\delta_+ = 0.75$. We find this tighter positive margin to be optimal under similarity regularization. However, we note that $\delta_+ = 0.75$ is likely to be too small when only the contrastive loss is used, since positive pairs are not encouraged to be more similar than a cosine similarity of 0.75.

H.6. Note on Stop Gradient

Note that we detach the normalization factors in Eq. 24. This is necessary because optimizing the size of the neighborhood is undesirable. On the contrary, we do not detach the normalization factor in Eq. 27. This is intentional even if somewhat un-intuitive. We show in Figure 17 that detaching $|\mathcal{R}_{k/2+\epsilon}(i)|$ in Eq. 27 increases R@1 on SOP when λ is high, but decreases R@1 in all other scenarios.

I. Additional Experimental Results

I.1. Experimental Results on Similarity Regularizer

In the main paper, we presented the similarity regularizer $\mathcal{L}_{\text{reg}}$ as complementary to the contextual similarity loss $\mathcal{L}_{\text{context}}$. However, there is no theoretical reason for limiting the regularizer to our contextual similarity framework. We show in Table 6 that the similarity regularizer offers an improvement in retrieval performance when combined with some baselines. In particular, the regularizer consistently improves R@1 performance of Triplet, Multi-Similarity, Smooth-AP, and ROADMAP

Table 6. Similarity Regularizer Results. Here, we experiment with naively adding our similarity regularizer to a subset of the baseline methods. The overall loss is $\gamma\mathcal{L}_{\text{reg}} + (1-\gamma)(\text{baseline loss})$ where $\gamma = 0.1$. We try two different values for $\tilde{s}$ and compare the difference in various performance metrics with the original baseline. The colored subscript denotes the increase or decrease in each metric when our similarity regularizer is added. Observe that in most cases, the regularizer improves the performance of the baseline or has negligible effect, with an appropriate $\tilde{s}$ value.

Cars

	mAP			mAP@R			R@1		
Method	No $\mathcal{L}_{\text{reg}}$	$\tilde{s}=0.25$	$\tilde{s}=0.3$	No $\mathcal{L}_{\text{reg}}$	$\tilde{s}=0.25$	$\tilde{s}=0.3$	No $\mathcal{L}_{\text{reg}}$	$\tilde{s}=0.25$	$\tilde{s}=0.3$
Contrastive	38.2	40.6 +2.4	39.4 +1.2	28.8	31.2 +2.4	30.0 +1.2	85.4	88.0 +2.6	87.1 +1.7
Triplet	34.9	39.2 +4.3	38.5 +3.6	24.7	29.9 +5.2	29.2 +4.5	77.6	87.8 +10.2	86.7 +9.1
NtXent	36.2	37.0 +0.8	35.8 -0.4	26.3	27.0 +0.7	26.0 -0.3	79.0	79.9 +0.9	78.7 -0.3
MS	43.1	42.7 -0.4	**43.6** +0.5	**33.7**	33.3 -0.4	34.1 +0.4	88.7	88.8 +0.1	**89.3** +0.6
Smooth-AP	37.5	38.9 +1.4	38.1 +0.6	27.5	28.8 +1.3	28.2 +0.7	81.1	82.1 +1.0	82.4 +1.3
ROADMAP	38.3	39.2 +0.9	39.6 +1.3	28.7	29.6 +0.9	30.0 +1.3	84.5	85.7 +1.2	85.7 +1.2
Ours	–	42.4	–	–	33.0	–	–	**89.3**	–

SOP

	mAP			mAP@R			R@1		
Method	No $\mathcal{L}_{\text{reg}}$	$\tilde{s}=0.25$	$\tilde{s}=0.3$	No $\mathcal{L}_{\text{reg}}$	$\tilde{s}=0.25$	$\tilde{s}=0.3$	No $\mathcal{L}_{\text{reg}}$	$\tilde{s}=0.25$	$\tilde{s}=0.3$
Contrastive	62.9	62.8 -0.1	62.7 -0.2	57.0	56.9 -0.1	56.8 -0.2	82.4	82.3 -0.1	82.3 -0.1
Triplet	63.1	64.4 +1.3	64.6 +1.5	56.4	57.8 +1.4	58.0 +1.6	82.0	83.1 +1.1	**83.3** +1.3
NtXent	60.1	60.3 +0.2	60.0 -0.1	53.4	53.7 +0.3	53.4 +0.0	79.7	80.0 +0.3	79.7 +0.0
MS	62.5	62.4 -0.1	62.6 +0.1	56.2	56.1 -0.1	56.3 +0.1	81.4	81.4 +0.0	81.8 +0.4
Smooth-AP	63.1	63.2 +0.1	63.5 +0.4	56.4	56.5 +0.1	56.8 +0.4	82.0	82.1 +0.1	82.3 +0.3
ROADMAP	64.4	64.4 +0.0	64.4 +0.0	58.2	58.2 +0.0	58.2 +0.0	83.1	83.1 +0.0	83.1 +0.0
Ours	–	**65.0**	–	–	**58.6**	–	–	**83.3**	–

iNaturalist

	mAP			mAP@R			R@1		
Method	No $\mathcal{L}_{\text{reg}}$	$\tilde{s}=0.25$	$\tilde{s}=0.3$	No $\mathcal{L}_{\text{reg}}$	$\tilde{s}=0.25$	$\tilde{s}=0.3$	No $\mathcal{L}_{\text{reg}}$	$\tilde{s}=0.25$	$\tilde{s}=0.3$
Contrastive	16.0	15.1 -0.9	15.5 -0.5	11.6	11.2 -0.4	11.5 -0.1	43.5	43.4 -0.1	43.7 +0.2
Triplet	12.1	13.3 +1.2	13.6 +1.5	7.9	9.8 +1.9	10.0 +2.1	35.4	41.4 +6.0	41.7 +6.3
NtXent	15.9	15.9 +0.0	15.9 +0.0	10.6	10.7 +0.1	10.6 +0.0	40.8	40.7 -0.1	40.7 -0.1
MS	16.6	16.6 +0.0	16.6 +0.0	11.9	11.9 +0.0	11.9 +0.0	44.9	44.9 +0.0	45.0 +0.1
Smooth-AP	16.5	16.6 +0.1	16.7 +0.2	11.3	11.4 +0.1	11.4 +0.1	42.7	43.1 +0.4	43.2 +0.5
ROADMAP	17.8	**18.1** +0.3	18.0 +0.2	12.7	**13.1** +0.4	13.0 +0.3	45.9	**47.1** +1.2	46.8 +0.9
Ours	–	17.1	–	–	12.4	–	–	46.2	–

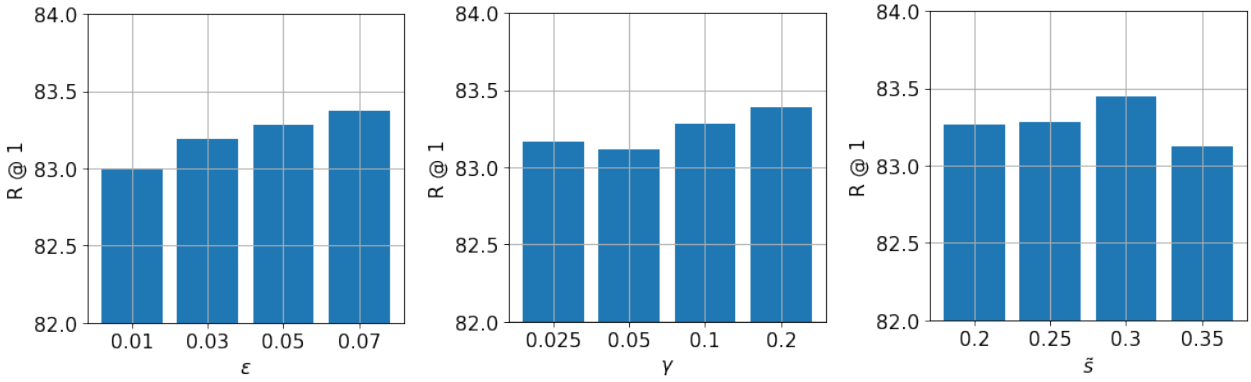

Figure 13. Hyperparameter tuning on SOP. We experiment with various values for ϵ, γ and $\tilde{s}$. The R@1 values are similar for the different hyperparameter choices.

losses across Cars, SOP and iNaturalist benchmarks. This is a promising result.

I.2. In-Shop Results

We offer R@k results for standard k values on the In-Shop dataset in Table 8. In-Shop ((Liu et al., 2016)) is a popular image retrieval benchmark. We use the standard train-test split: we use 25,882 images from 3,997 classes for training; for testing, we use a query set with 14,218 images and a gallery set (aka. reference set) with 12,612 images, both from 3,985 classes. We use the same augmentation and batch size as SOP. An epoch is defined as 600 batches.

I.3. Average Precision Results for Table 4

We offer Average Precision (AP) results in Table 7. We present results for two flavors of AP ((McFee & Lanckriet, 2010)): mAP and mAP@R. mAP is the precision at k, averaged across correctly retrieved samples, averaged across the query set. mAP@R ((Musgrave et al., 2020a)) only averages across retrievals up to the number of positive pairs in the gallery set. Hence, mAP@R is always smaller than mAP. The two versions of AP are equivalent bases for comparison. In the following equations, $\mathcal{X}_i^+$ denotes the set of samples in the gallery set with the same label as query i and $\mathcal{X}_i^-$ denotes the set of samples in the gallery set with a different label than query i. Prec@k denotes the precision at k, which is the percentage of samples ranked less than or equal to k with the same label as the query.

$$\text{mAP}_i = \frac{1}{|\mathcal{X}_i^+|} \sum_{k=1}^{|\mathcal{X}_i^+|+|\mathcal{X}_i^-|} \text{Prec@}k \; \mathbb{1}[k \in \mathcal{X}_i^+]$$

$$\text{mAP@R}_i = \frac{1}{|\mathcal{X}_i^+|} \sum_{k=1}^{|\mathcal{X}_i^+|} \text{Prec@}k \; \mathbb{1}[k \in \mathcal{X}_i^+]$$

I.4. Comparison on Hierarchical Retrieval Metrics

Over-reliance on binary supervision is a long-standing problem in metric learning which motivates our contextual loss. Another interesting line of work uses hierarchical labels during training to mitigate this over-reliance (Sun et al. (2021), Zheng et al. (2022), and Ramzi et al. (2022)). These works fall under the umbrella of "dynamic metric learning" and "hierarchical metric learning". One potential drawback of these works is their requirement for hierarchical labels, which may not always be available. Broadly speaking, hierarchical metric learning losses enforce a larger penalty on mistakes in discriminating between labels that are farther apart in the hierarchy during training. The resulting embedding space is more consistent in the sense that most retrieval mistakes are between labels close together in the hierarchy. For example, if the query is an image of a bird, any incorrectly retrieved images are likely to be birds from a closely related species, rather than images of other animals.

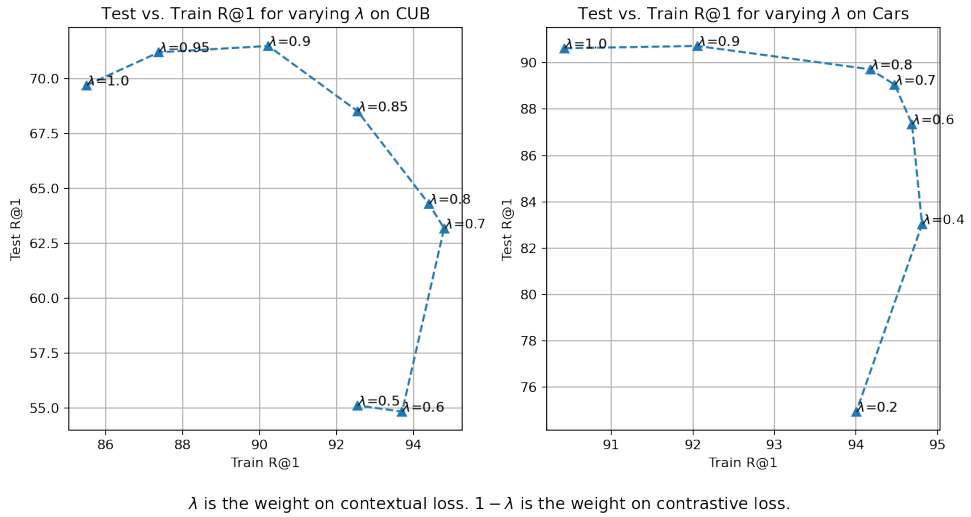

Figure 14. Test R@1 vs. train R@1 accuracy for varying λ ($\gamma = 0$). This figure shows that adding the standard contrastive loss (lowering λ) increases the R@1 on training data. This also increases the test R@1 up to a point (approximately $\lambda = 0.9$), before overfitting.

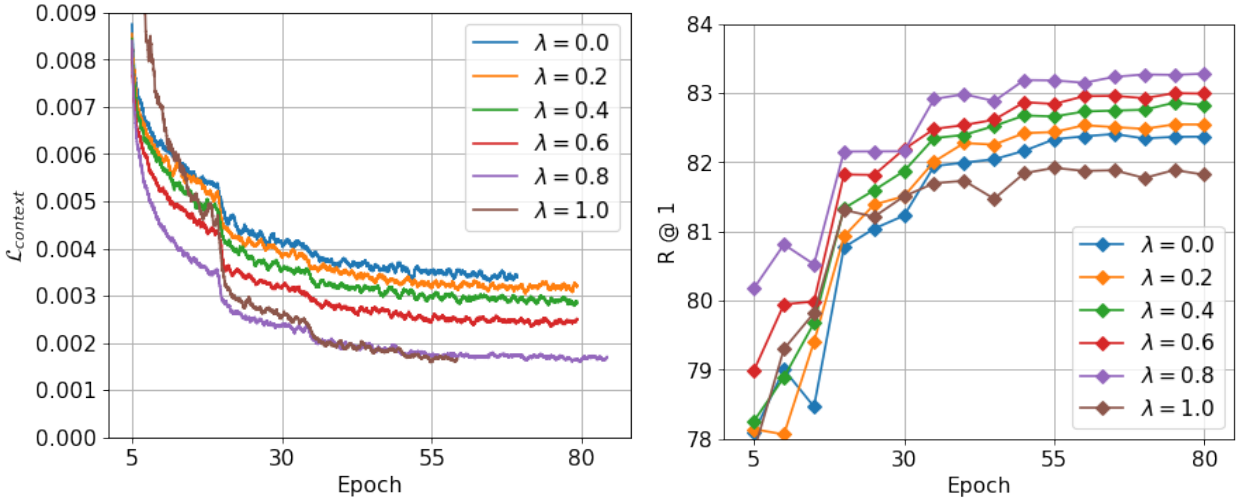

Figure 15. Plot of contextual similarity loss $\mathcal{L}_{\text{context}}$ with varying λ. Each color is a different λ. Results are on the SOP dataset. Observe that the contrastive loss implicitly minimizes the contextual loss (blue line). Also observe that in general, a lower contextual loss corresponds to a higher test R@1. This shows that our loss function is a reasonable objective. Note that the plot on the left plots $\mathcal{L}_{\text{context}}$, not the complete training loss. $\gamma = 0$.

Table 7. mAP and mAP@R Results. We use ResNet-50 with an embedding size of 512 for all experiments. We re-run all baselines in this Table using the implementation by (Musgrave et al., 2020b). Standard deviations are based on three trials with the same train-test split.

	CUB		Cars		SOP		mini-iNaturalist	
Method	mAP	mAP@R	mAP	mAP@R	mAP	mAP@R	mAP	mAP@R
Contrastive	36.6 ± 0.4	26.6 ± 0.5	38.2 ± 0.4	28.8 ± 0.5	62.9 ± 0.1	57.0 ± 0.1	16.0 ± 0.1	11.6 ± 0.0
Triplet	36.9 ± 0.4	26.5 ± 0.2	34.9 ± 0.8	24.7 ± 0.7	63.1 ± 0.0	56.4 ± 0.0	12.1 ± 0.0	7.9 ± 0.0
NtXent	36.3 ± 0.3	26.0 ± 0.3	36.2 ± 0.7	26.3 ± 0.7	60.1 ± 0.1	53.4 ± 0.1	15.9 ± 0.0	10.6 ± 0.0
MS	38.0 ± 0.1	27.7 ± 0.1	**43.1** ± 0.3	**33.7** ± 0.3	62.5 ± 0.0	56.2 ± 0.1	16.6 ± 0.0	11.9 ± 0.0
Fast-AP	34.4 ± 0.8	24.4 ± 0.7	32.7 ± 0.0	23.5 ± 0.0	60.7 ± 0.1	54.2 ± 0.1	13.9 ± 0.1	9.1 ± 0.1
Smooth-AP	36.8 ± 0.4	26.4 ± 0.4	37.5 ± 0.3	27.5 ± 0.2	63.1 ± 0.2	56.4 ± 0.2	16.5 ± 0.0	11.3 ± 0.0
ROADMAP	37.7 ± 0.1	27.5 ± 0.2	38.3 ± 0.4	28.7 ± 0.4	64.4 ± 0.1	58.2 ± 0.1	**17.8** ± 0.0	**12.7** ± 0.0
Ours	**38.3** ± 0.2	**28.0** ± 0.2	42.4 ± 0.4	33.0 ± 0.3	**65.0** ± 0.1	**58.6** ± 0.1	17.1 ± 0.0	12.4 ± 0.0

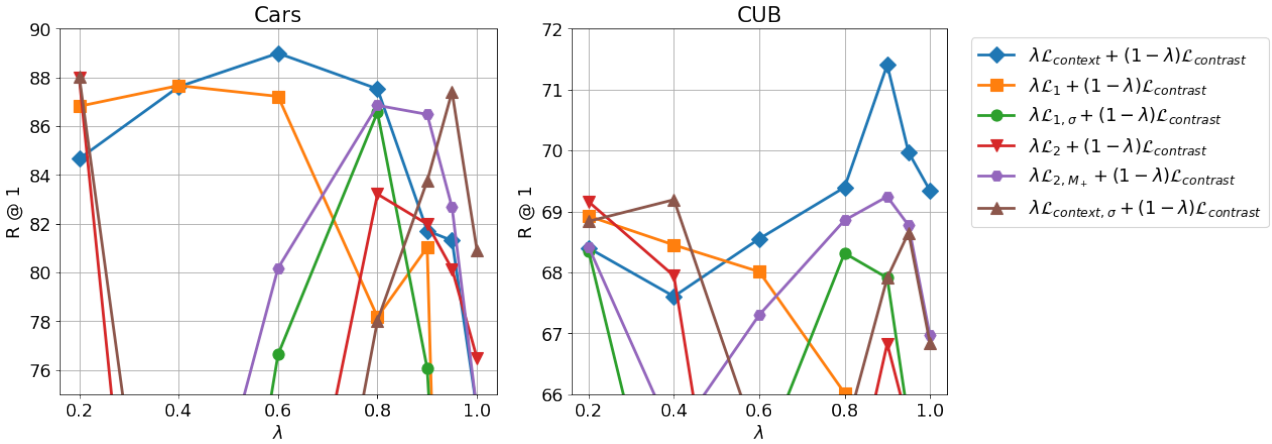

Figure 16. Ablation results on CUB and Cars. The blue line represents test R@1 of our framework for varying λ with $\gamma = 0$. The other colors represent various modifications or simplifications of the loss function (reference Table 5 and Section F for full description). We perform each ablation experiment with varying λ since the optimal λ is not constant across all experiments. Clearly, the modified versions of $\mathcal{L}_{\text{context}}$ are sub-optimal.

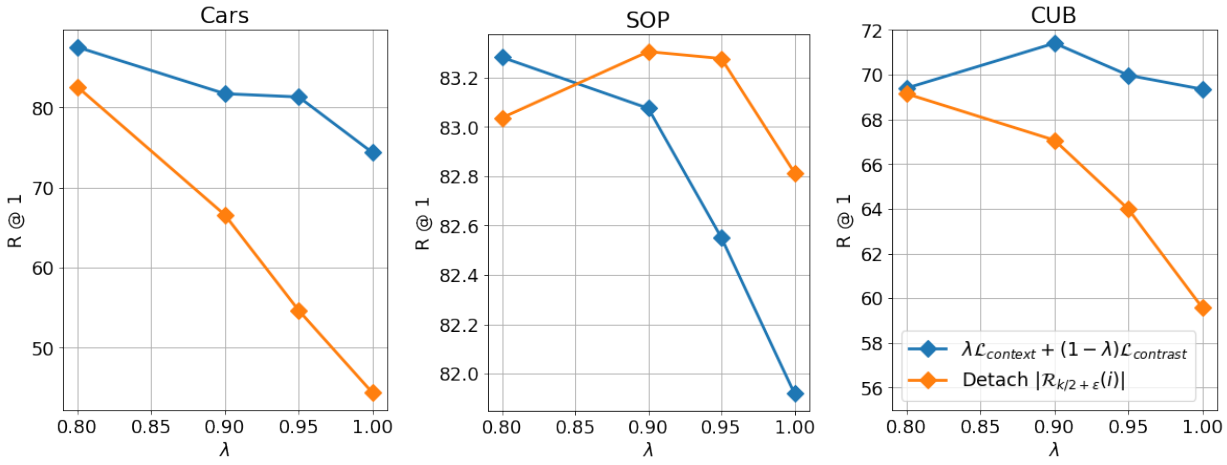

Figure 17. Investigating the effect of detaching $|\mathcal{R}_{k/2+\epsilon}(i)|$ in Eq. 27. The blue line shows R@1 of our framework with varying λ. $\gamma = 0$. The orange line shows R@1 after detaching $|\mathcal{R}_{k/2+\epsilon}(i)|$. Overall, detaching this normalization factor is undesirable.

Table 8. In-Shop Results. We use ResNet-50 with an embedding size of 512 for all experiments. † indicates results reported by the original authors; we re-run all other baselines using the implementation by (Musgrave et al., 2020b). Our R@1 performance is better than all baselines. We are at least comparable to baselines on other R@k metrics.

	In-Shop					
Method	R@1	R@10	R@20	R@30	R@40	R@50
Contrastive	90.1	97.4	98.3	98.6	98.8	98.9
Triplet	90.2	98.0	98.7	**99.0**	**99.2**	**99.3**
NtXent	89.3	97.6	98.3	98.7	98.9	99.0
MS	86.9	96.1	97.3	97.9	98.1	98.3
N-Softmax†	88.6	97.5	98.4	98.8	–	–
Proxy NCA ++†	90.4	**98.1**	**98.8**	**99.0**	**99.2**	–
Fast-AP	89.7	97.5	98.3	98.6	98.8	98.9
Smooth-AP	90.2	97.9	98.7	**99.0**	**99.2**	99.2
ROADMAP	90.4	97.6	98.3	98.6	98.8	99.0
Ours	**90.7**	97.8	98.5	98.9	99.1	99.2

Table 9. Hierarchical Retrieval Results. We use ResNet-50 with an embedding size of 512 for all experiments. We re-run all baselines in this table using the implementation by (Musgrave et al., 2020b). Image size is 224x224, and the experimental settings follow the settings presented in Section 5 of the main paper. The results in this table are from one random trial. "fine R@1" indicates the R@1 at the fine-grained label level (which is the same R@1 as the rest of the paper). On CUB, there are two higher levels in the label hierarchy: the family and order that the bird species belongs to, indicated as "mid" and "coarse", resp. On Cars, the coarse label is the make of the vehicle. On SOP, the coarse label is the category of the product. We follow the hierarchical labels used in (Chang et al., 2021) for CUB and Cars; we use the coarse labels given by the original SOP dataset.

	CUB				Cars			SOP		
Method	fine R@1	fine AP	mid AP	coarse AP	fine R@1	fine AP	coarse AP	fine R@1	fine AP	coarse AP
Contrastive	66.2	35.9	52.0	85.7	81.1	33.4	39.8	80.7	60.1	13.0
Roadmap	66.2	36.0	53.8	88.1	83.5	37.3	41.9	82.1	62.6	13.9
Triplet	65.0	34.5	53.8	**89.8**	89.3	43.3	**42.7**	81.8	62.5	15.0
MS+miner	69.4	37.4	53.5	88.5	90.7	42.7	39.4	82.1	63.2	13.5
Proxy Anchor	68.0	36.6	53.9	89.1	88.8	39.0	38.6	79.8	59.4	16.2
Proxy NCA	65.7	34.9	**55.5**	88.0	88.0	37.6	38.5	78.9	58.6	**16.8**
Contextual (ours)	**72.7**	**40.3**	54.3	86.4	**91.0**	**43.4**	39.1	**82.7**	**63.7**	13.5

The contextual loss as presented in this paper does not use hierarchical labels, since our focus is advancing the state-of-the-art in fine-grained retrieval, irrespective of coarse-grained retrieval. Keeping this in mind, we offer results on hierarchical retrieval metrics in Table 9. From Table 9, we conclude that our contextual loss always achieves the highest fine-grained performance, but under-performs baselines at higher levels in the hierarchy. This trade-off in fine vs. coarse retrieval is consistent with results from HAPPIER (Ramzi et al., 2022). We hope that these results will be helpful for future work on adapting the contextual loss to the hierarchical retrieval setting.

I.5. Contextual Similarity Re-ranking at Test Time

Using contextual similarity for re-ranking at test time is a common trick used to improve recall accuracy. However, this is not a common evaluation setting, so we omitted it from the main paper. We offer R@1 results with contextual re-ranking in Table 10. The contextual re-ranking procedure is as follows. After training, we calculate the cosine similarity between all pairs of samples in the test set. Denote this as s_{ij}, for each pair of test samples i and j. We then calculate a *simplified version* of the contextual similarity $\tilde{w}_{ij}$ between each pair of test samples i and j using Equations 15 and 16 in Appendix E. Note that $s_{ij} \in [-1, 1]$ while $\tilde{w}_{ij} \in [0, 1]$, so we need to rectify s_{ij} to occupy the same interval as $\tilde{w}_{ij}$. To this end, define $\tilde{s}_{ij} = \exp(2s_{ij} - 2)$. For retrieval, we use a weighted sum of the rectified cosine similarity and the contextual similarity: $\beta\tilde{w}_{ij} + (1 - \beta)\tilde{s}_{ij}$.

Table 10. Contextual re-ranking R@1 results. $\beta = 0.1$ is weight on contextual similarity. We experiment with different values for k. We use ResNet-50 with an embedding size of 512 for all experiments. We re-run all baselines in this table using the implementation by (Musgrave et al., 2020b). Image size is 224x224, and the experimental settings follow the settings presented in Section 5 of the main paper. The results in this table are from one random trial.

	CUB			Cars			SOP	
Method	no re-rank	$k = 16$	$k = 32$	no re-rank	$k = 16$	$k = 32$	no re-rank	$k = 4$
Contrastive	66.2	66.6	66.8	81.1	81.2	80.8	80.7	81.0
Roadmap	66.2	67.4	66.9	83.5	83.5	83.3	82.1	82.3
Triplet	65.0	66.6	65.8	89.3	90.0	89.9	81.8	82.2
MS+miner	69.4	71.1	70.7	90.7	91.2	91.0	82.1	82.4
Proxy Anchor	68.0	69.1	69.0	88.8	89.3	89.1	79.8	80.1
Proxy NCA	65.7	67.2	66.9	88.0	88.5	88.5	78.9	79.1
Contextual (ours)	**72.7**	**74.3**	**73.9**	**91.0**	**91.4**	**91.1**	**82.7**	**82.9**

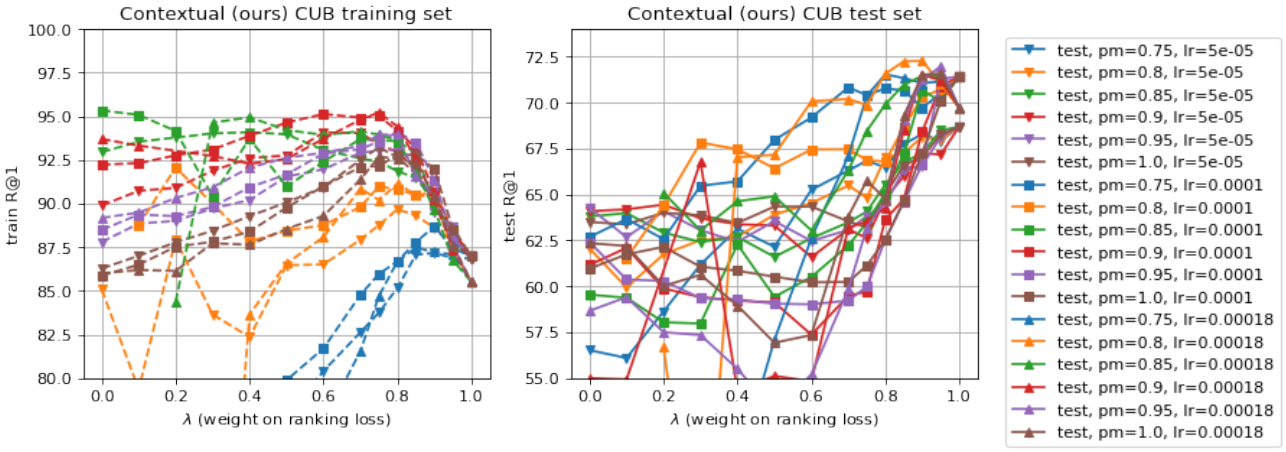

Figure 18. CUB R@1 results for our contextual loss function, with varying λ, learning rate, and positive margin on contrastive loss. We plot both train and test R@1 results. "lr" and "pm" in the legend stand for learning rate and positive margin δ_+, respectively.

Table 10 shows that all methods benefit from contextual re-ranking, and our contextual loss remains the best method across the three benchmarks. Embeddings optimized by the contextual loss do not benefit more from contextual re-ranking than baselines (i.e. the relative increase in R@1 with contextual re-ranking is about the same across all methods). There is no theory to suggest that optimizing contextual similarity at train time results in better contextual similarity re-ranking at test time, and we make no claims regarding this.

I.6. Additional Comparisons to AP Surrogates

Figures 18 through 29 present a comparison between our contextual loss function and various AP surrogates. The goal of these figures is to show that our contextual loss function is definitively better at ranking than AP surrogates. We optimize a convex combination of the ranking loss and standard contrastive loss. We present results for different choices of learning rates, λ (weight on ranking loss), and positive margin on the contrastive loss, since the optimal hyperparameters vary between methods. We include results for all three benchmarks. The AP surrogates included in this comparison are: Roadmap (Ramzi et al., 2021), Smooth-AP (Brown et al., 2020), Softbin-AP (Revaud et al., 2019), and Blackbox-AP (Rolínek et al., 2020).

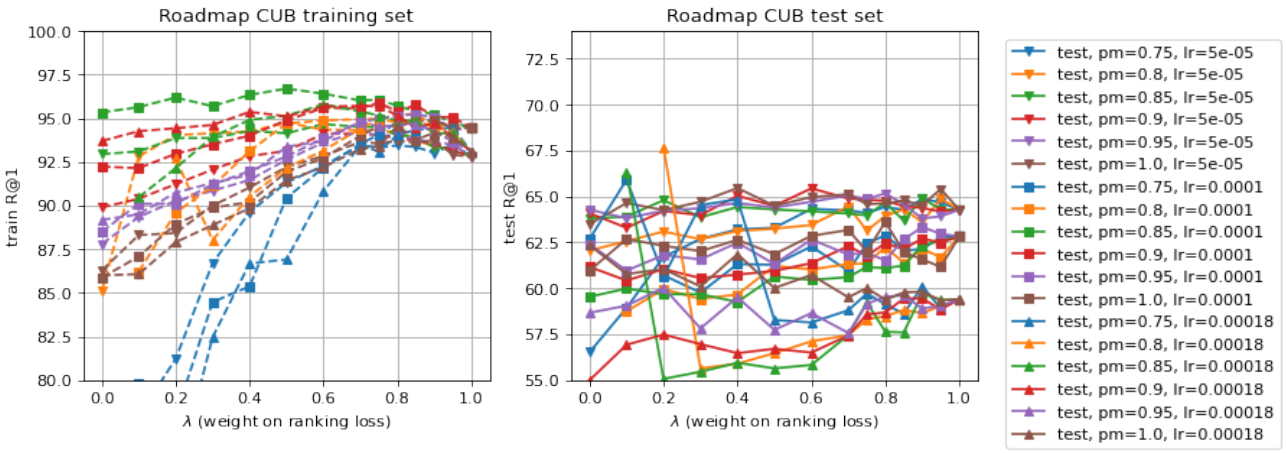

Figure 19. CUB R@1 results for Roadmap, with varying λ, learning rate, and positive margin on contrastive loss. We plot both train and test R@1 results. "lr" and "pm" in the legend stand for learning rate and positive margin δ_+, respectively.

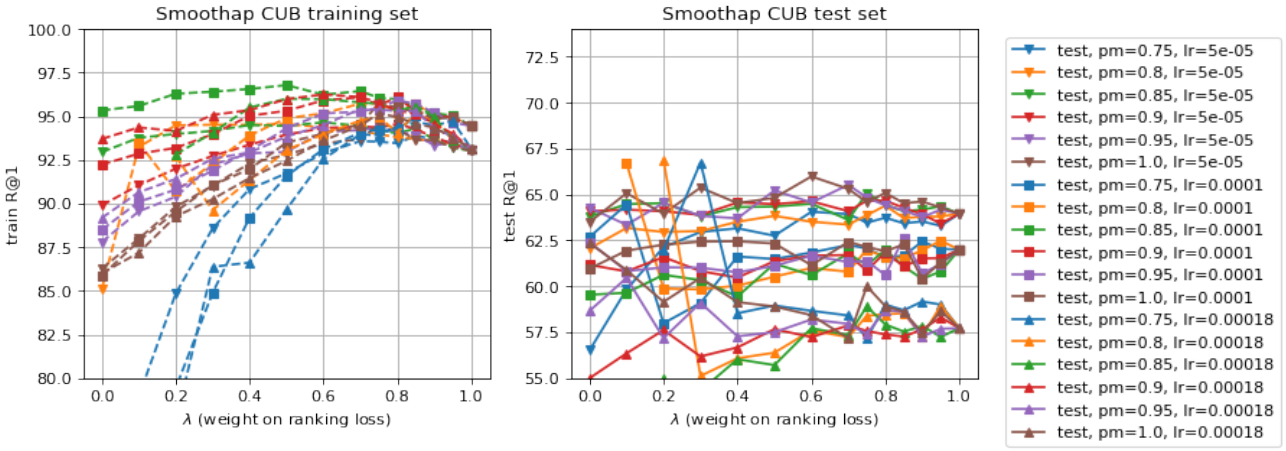

Figure 20. CUB R@1 results for Smooth-AP, with varying λ, learning rate, and positive margin on contrastive loss. We plot both train and test R@1 results. "lr" and "pm" in the legend stand for learning rate and positive margin δ_+, respectively.

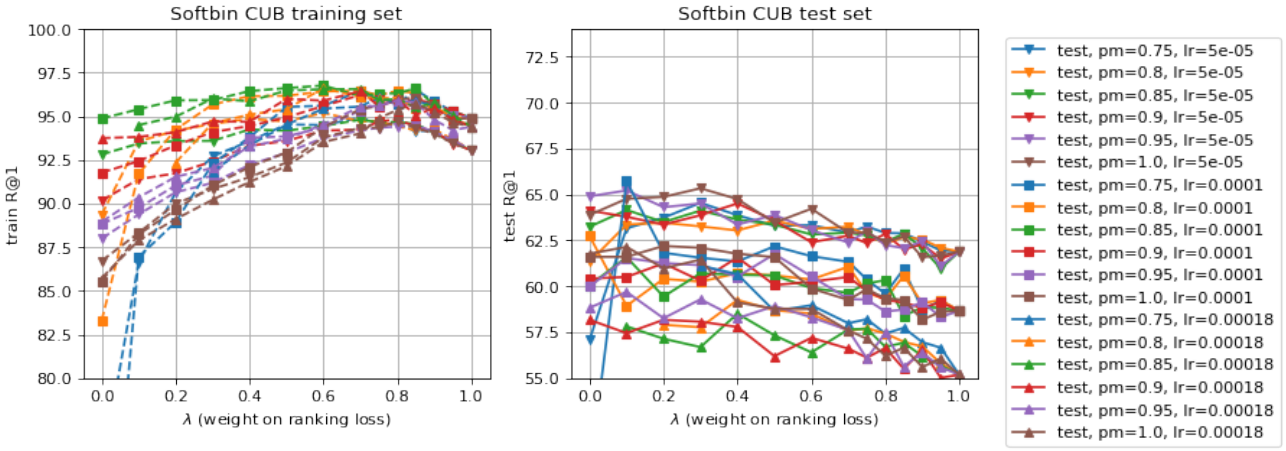

Figure 21. CUB R@1 results for Softbin-AP, with varying λ, learning rate, and positive margin on contrastive loss. We plot both train and test R@1 results. "lr" and "pm" in the legend stand for learning rate and positive margin δ_+, respectively.

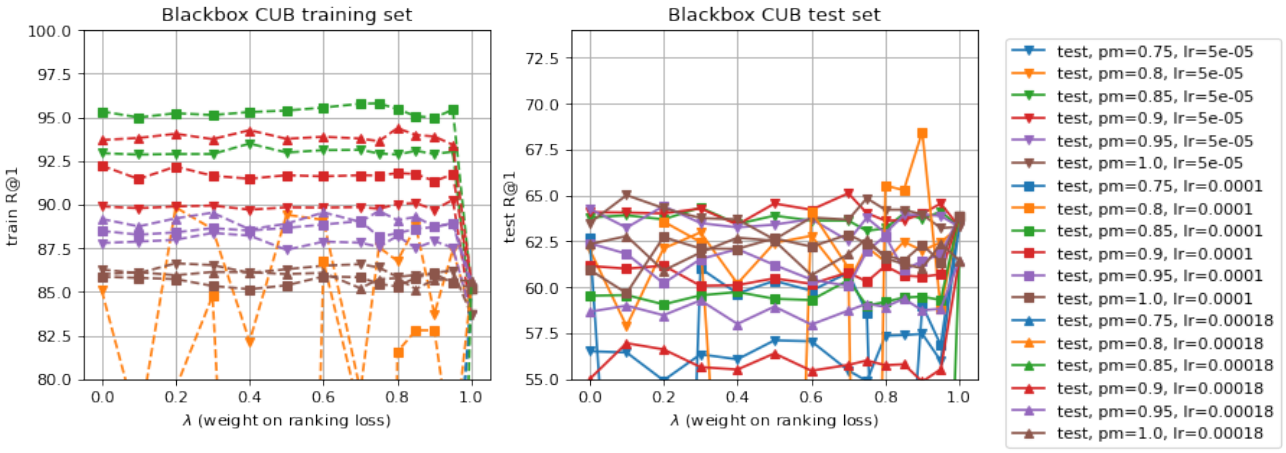

Figure 22. CUB R@1 results for Blackbox-AP, with varying λ, learning rate, and positive margin on contrastive loss. We plot both train and test R@1 results. "lr" and "pm" in the legend stand for learning rate and positive margin δ_+, respectively.

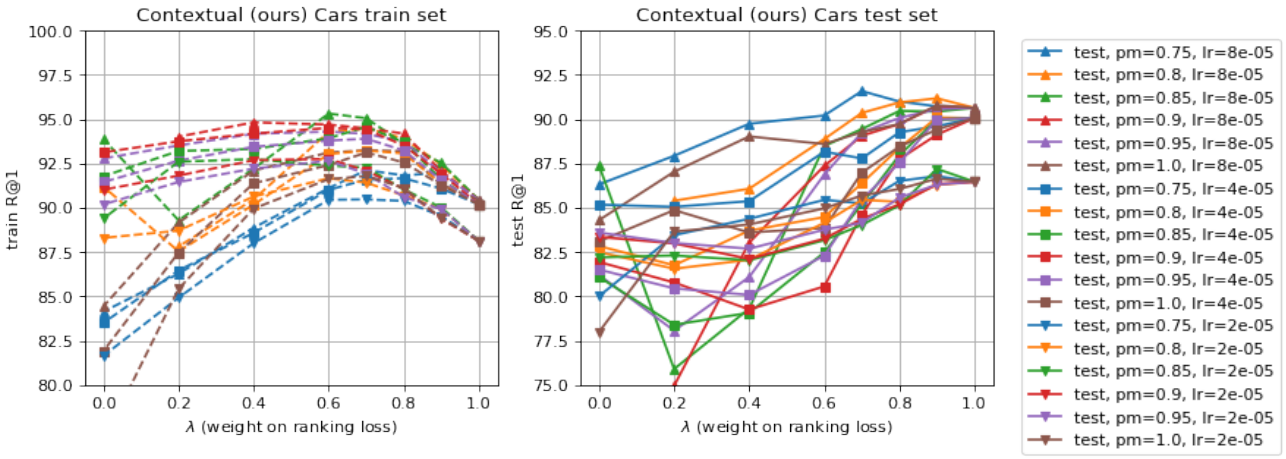

Figure 23. Cars R@1 results for our contextual loss function, with varying λ, learning rate, and positive margin on contrastive loss. We plot both train and test R@1 results. "lr" and "pm" in the legend stand for learning rate and positive margin δ_+, respectively.

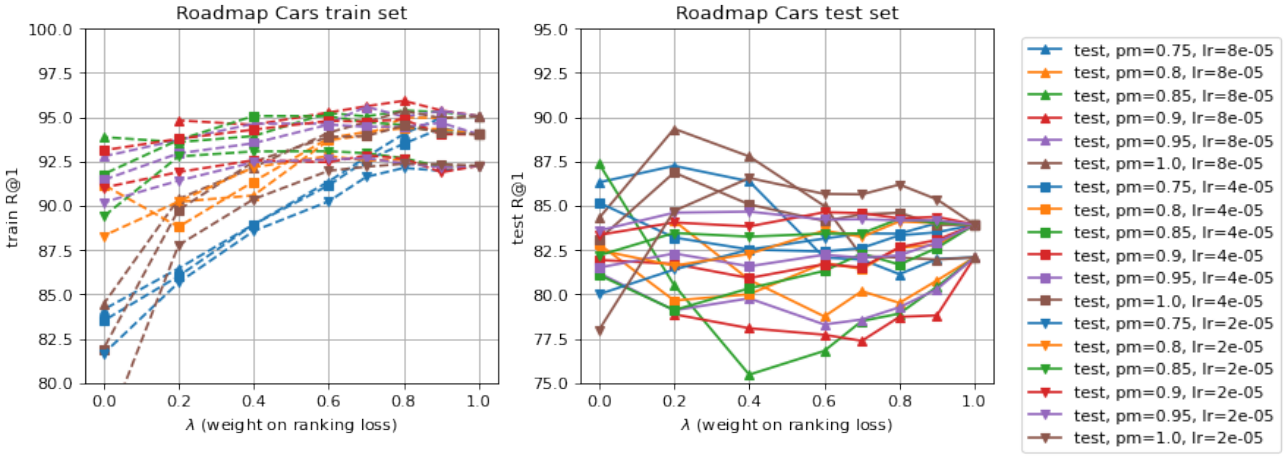

Figure 24. Cars R@1 results for Roadmap, with varying λ, learning rate, and positive margin on contrastive loss. We plot both train and test R@1 results. "lr" and "pm" in the legend stand for learning rate and positive margin δ_+, respectively.

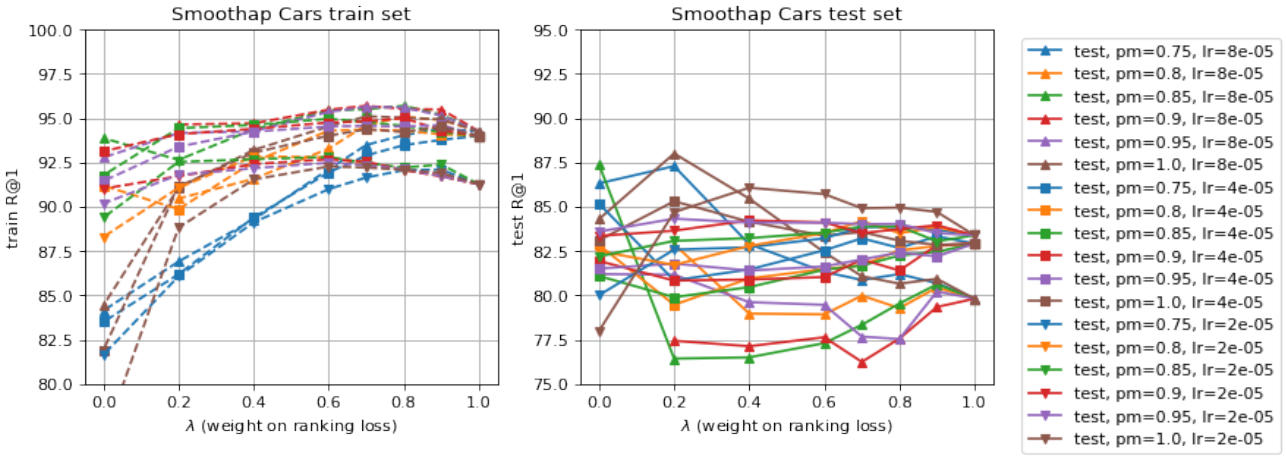

Figure 25. Cars R@1 results for Smooth-AP, with varying λ, learning rate, and positive margin on contrastive loss. We plot both train and test R@1 results. "lr" and "pm" in the legend stand for learning rate and positive margin δ_+, respectively.

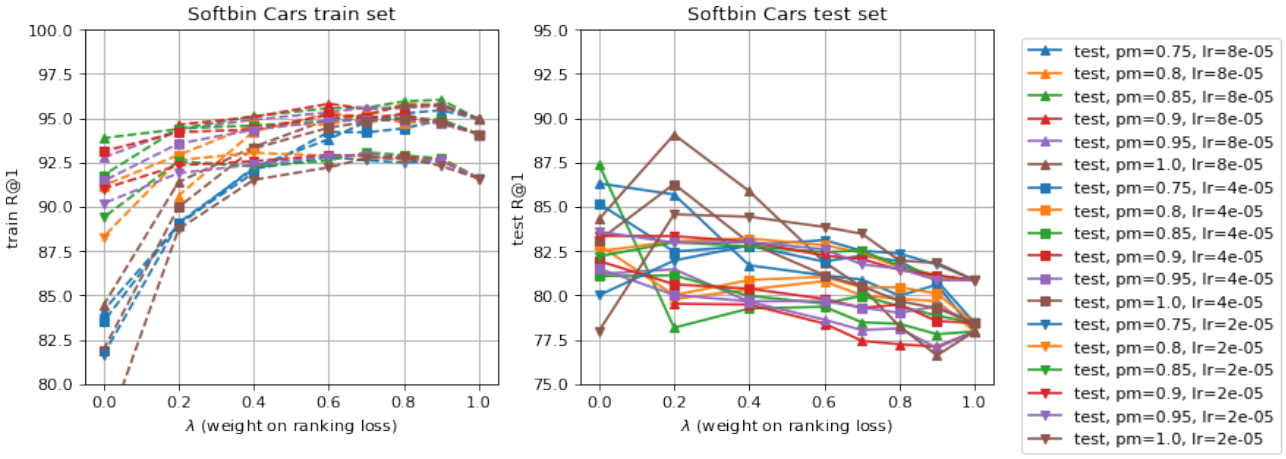

Figure 26. Cars R@1 results for Softbin-AP, with varying λ, learning rate, and positive margin on contrastive loss. We plot both train and test R@1 results. "lr" and "pm" in the legend stand for learning rate and positive margin δ_+, respectively.

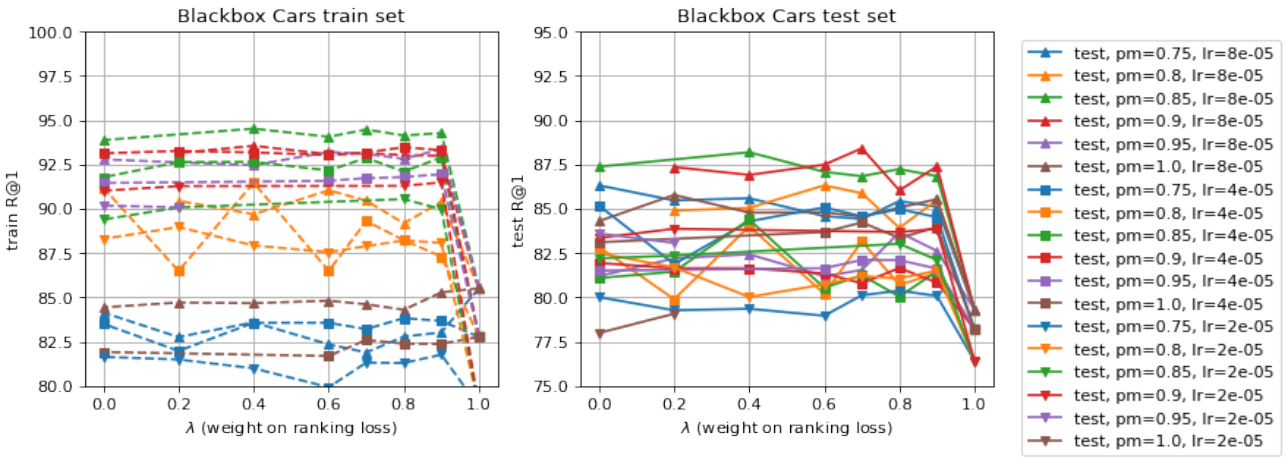

Figure 27. Cars R@1 results for Blackbox-AP, with varying λ, learning rate, and positive margin on contrastive loss. We plot both train and test R@1 results. "lr" and "pm" in the legend stand for learning rate and positive margin δ_+, respectively.

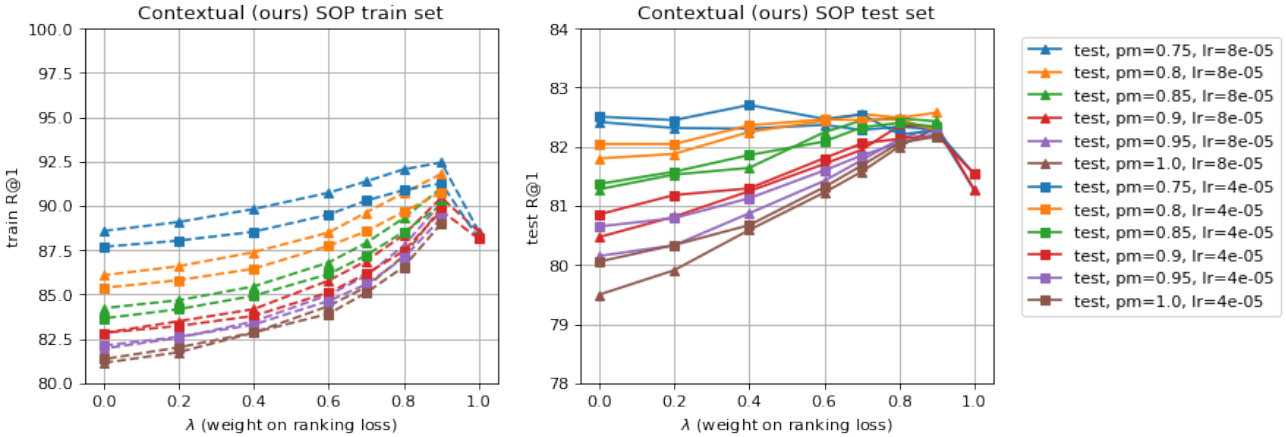

Figure 28. SOP R@1 results for our contextual loss function, with varying λ, learning rate, and positive margin on contrastive loss. We plot both train and test R@1 results. "lr" and "pm" in the legend stand for learning rate and positive margin δ_+, respectively.

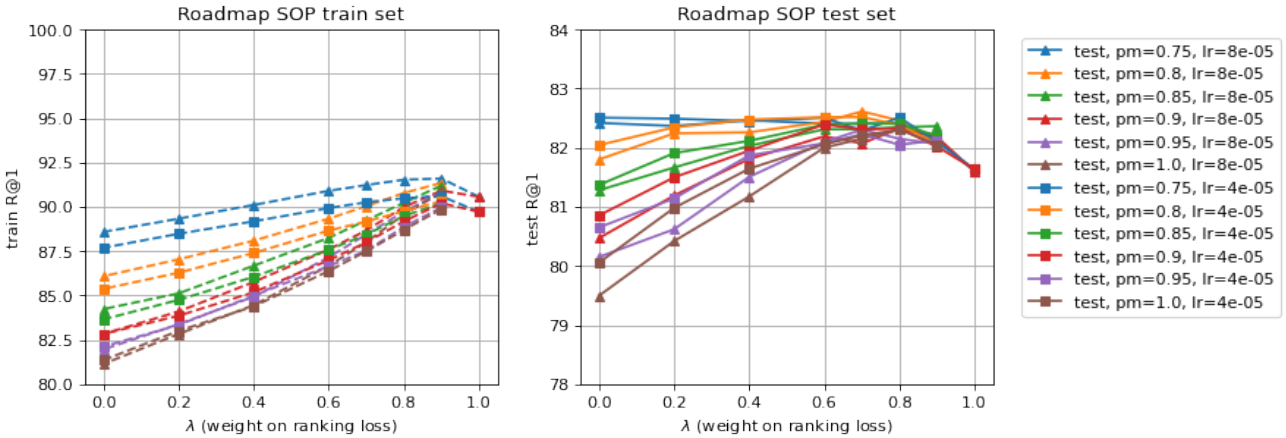

Figure 29. SOP R@1 results for Roadmap, with varying λ, learning rate, and positive margin on contrastive loss. We plot both train and test R@1 results. "lr" and "pm" in the legend stand for learning rate and positive margin δ_+, respectively.