Generalized Sum Pooling for Metric Learning

Yeti Z. Gürbüz[†] Ozan Şener A. Aydın Alatan
RSiM, TU Berlin Intel Labs OGAM and METU

Abstract

A common architectural choice for deep metric learning is a convolutional neural network followed by global average pooling (GAP). Albeit simple, GAP is a highly effective way to aggregate information. One possible explanation for the effectiveness of GAP is considering each feature vector as representing a different semantic entity and GAP as a convex combination of them. Following this perspective, we generalize GAP and propose a learnable generalized sum pooling method (GSP). GSP improves GAP with two distinct abilities: i) the ability to choose a subset of semantic entities, effectively learning to ignore nuisance information, and ii) learning the weights corresponding to the importance of each entity. Formally, we propose an entropy-smoothed optimal transport problem and show that it is a strict generalization of GAP, i.e.*, a specific realization of the problem gives back GAP. We show that this optimization problem enjoys analytical gradients enabling us to use it as a direct learnable replacement for GAP. We further propose a zero-shot loss to ease the learning of GSP. We show the effectiveness of our method with extensive evaluations on 4 popular metric learning benchmarks. Code is available at: GSP-DML Framework*

1. Introduction

Distance metric learning (DML) addresses the problem of finding an embedding function such that the semantically similar samples are embedded close to each other while the dissimilar ones are placed relatively apart in the Euclidean sense. Although the prolific and diverse literature of DML includes various architectural designs [9, 24, 33], loss functions [39], and data-augmentation techniques [47, 55], many of these methods have a shared component: a convolutional neural network (CNN) followed by a global pooling layer, mostly global average pooling (GAP) [39].

Common folklore to explain the effectiveness of GAP is considering each pixel of the CNN feature map as corresponding to a separate semantic entity [14]. For example, spatial extent of one pixel can correspond to a *"tire"* object making the resulting feature a representation for *"tireness"* of the image. If this explanation is correct, the representation space defined via output of GAP is a convex combination of semantically independent representations defined by each pixel in the feature map. Although this folklore is later empirically studied in [64, 70, 71, and references therein] and further verified for classification in [13, 62], its algorithmic implications are not clear. If each feature is truly representing a different semantic entity, should we really average over all of them? Surely, some classes belong to the background and should be discarded as nuisance variables. Moreover, is uniform average of them the best choice? Aren't some classes more important than others? In this paper, we try to answer these questions within the context of metric learning. We propose a learnable and generalized version of GAP which learns to choose the subset of the semantic entities to utilize as well as weights to assign them while averaging.

In order to generalize the GAP operator to be learnable, we re-define it as a solution of an optimization problem. We let the solution space to include 0-weight effectively enabling us to choose subset of the features as well as carefully regularize it to discourage degenerate solution of using all the features. Crucially, we rigorously show that the original GAP is a specific case of our proposed optimization problem for a certain realization. Our proposed optimization problem closely follows optimal transport based *top-k* operators [6] and we utilize its literature to solve it. Moreover, we present an algorithm for an efficient computation of the gradients over this optimization problem enabling learning. A critical desiderata of such an operator is choosing subset of features which are discriminative and ignoring the background classes corresponding to nuisance variables. Although supervised metric learning losses provide guidance for seen classes, they carry no such information to generalize the behavior to unseen classes. To enable such a behavior, we adopt a *zero-shot prediction loss*

[†]Affiliated with OGAM-METU during the research.

as a regularization term which is built on expressing the class label embeddings as a convex combination of attribute embeddings [7, 62].

In order to validate the theoretical claims, we design a synthetic empirical study. The results confirm that our pooling method chooses better subsets and improve generalization ability. Moreover, our method can be applied with any DML loss as GAP is a shared component of them. We applied our method on 6 DML losses and test on 4 datasets. Results show consistent improvements with respect to direct application of GAP as well as other pooling alternatives.

2. Related Work

Our contributions. Briefly, our contributions include that *i*) we introduce a general formulation for weighted sum pooling, *ii*) we formulate local feature selection as an optimization problem which admits closed form gradient expression without matrix inversion, and *iii*) we propose a meta-learning based zero-shot regularization term to explicitly impose unseen class generalization to the DML problem. We now discuss the works which are most related to ours.

Distance Metric Learning (DML). Primary thrusts in DML include *i*) tailoring pairwise loss terms [39] that enforces the desired intra- and inter-class proximity constraints, *ii*) pair mining [47], *iii*) generating informative samples [12, 27, 34, 55], and *iv*) augmenting the mini-batches with virtual embeddings called *proxies* [53,58]. To improve generalization; learning theoretic ideas [8, 15, 31], disentangling class-discriminative and class-shared features [33, 45], intra-batch feature aggregation [32,52], ranking surrogates [43], and further regularization terms [4, 20, 25, 48, 65] are utilized. To go beyond of a single model, ensemble [24,50,63,68,69] and multi-task based approaches [38, 46] are also used. Different to them, we propose a learnable pooling method generalizing GAP, a shared component of all of the mentioned works. Hence, our work is orthogonal to all of these and can be used jointly with any of them.

Prototype-based pooling. Most related to ours are trainable VLAD [2, 66] and optimal transport based aggregation [28, 37]. Such methods employ similarities to the prototypes to form a vector of aggregated local features for each prototype and build ensemble of representations. Although their embeddings enjoy $\ell 2$ metric for similarity, they typically have very large sizes limiting their applicability for DML. Recently, reducing the dimension of VLAD embedding via non-linear transforms is addressed for DML [22]. Nevertheless, such methods still map a set of features to another set of features without discarding any and do not provide a natural way to aggregate the class-discriminative subset of the features. On the contrary, our pooling machine effectively enables learning to select discriminative features and maps a set of features to a single feature that is distilled from nuisance information.

Attention-based pooling. CroW [21], Trainable-SMK [54], and CBAM [59] reweight the CNN features before pooling. They build on feature magnitude based saliency, assuming that the backbone functions must be able to zero-out nuisance information. Yet, such a requirement is restrictive for the parameter space and annihilation of the non-discriminative information might not be feasible in some problems. Similarly, attention-based weighting methods DeLF [41], GSoP [11] do not have explicit control on feature selection behavior and might result in poor models when jointly trained with the feature extractor [41]. Differently, our method unifies attention-based feature masking practices (*e.g.*, *convolution*, *correlation*) with an efficient-to-solve optimization framework and lets us do away with engineered heuristics in obtaining the masking weights (*e.g.*, *normalization*, *sigmoid*, *soft-plus*) without restricting the solution space.

Optimal transport (OT) based operators. OT distance [5] to match local features is used as the DML distance metric instead of $\ell 2$ in [67]. Despite effective, replacing $\ell 2$ with OT increases memory cost for image representation as well as computation cost for the distance computation. Different to them, we shift OT based computation in pooling (*i.e.*, feature extraction) stage while having OT's merits and hence, do not affect the memory and computation costs of the inference by sticking to $\ell 2$ metric. Moreover, our feature selection and aggregation formulation has close relation to OT [5] based top-k [61], ranking [6] and aggregation [28, 37, 40] operators which are not effectively applied to DML before. Their aggregation is built on concatenating the feature ensembles resulting in very large embedding sizes. What makes our method different is the unique way we formulate the feature selection problem to fuse aggregation into it (*see Appendix for technical details*). Our selection operator allows computationally appealing and matrix inversion free gradient computation unlike its OT based counterparts [36].

3. Preliminaries

Consider the data distribution $p_{\mathcal{X}\times\mathcal{Y}}$ over $\mathcal{X}\times\mathcal{Y}$ where $\mathcal{X}$ is the space of data points and $\mathcal{Y}$ is the space of labels. Given *iid.* samples from $p_{\mathcal{X}\times\mathcal{Y}}$ as $\{(x_i, y_i)\}$, distance metric learning problem aims to find the parameters θ of an embedding function $e(\cdot;\theta) : \mathcal{X} \to \mathbb{R}^d$ such that the Euclidean distance in the space of embeddings is consistent with the label information where d is the embedding dimension. More specifically, $\|e(x_i;\theta) - e(x_j;\theta)\|_2$

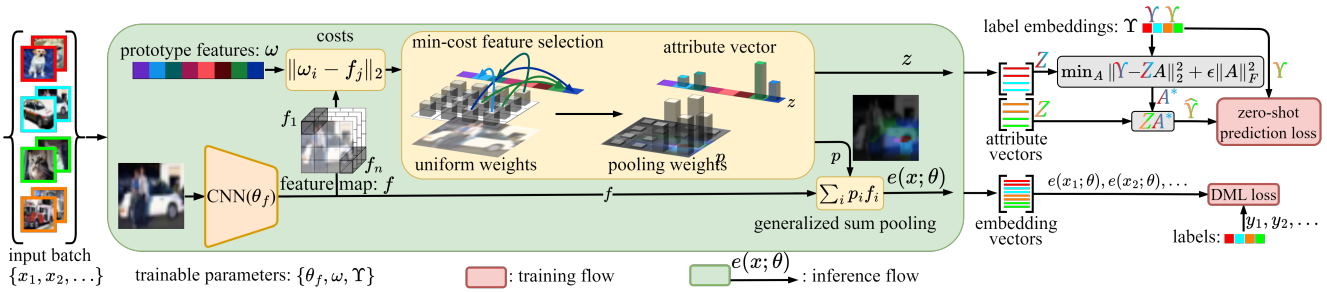

Figure 1: Sketch of the method, where $Z{=}[z_i]_i$ (4.3) and GSP vectors (4.1) are coloured w.r.t. their class label.

is small whenever $y_i = y_j$, and large whenever $y_i \neq y_j$. In order to enable learning, this requirement is represented via loss function $l((x_i, y_i), (x_j, y_j); \theta)$ (*e.g.*, *contrastive* [60], *triplet* [51], *multi-similarity* [57]).

The typical learning mechanism is gradient descent of an empirical risk function defined over a batch of data points B. To simplify notation throughout the paper, we will use $b = \{b(i) \mid x_i, y_i \in B\}_i$ to index the samples in a batch. Then, the typical empirical risk function is defined as:

$$\mathcal{L}_{\text{DML}}(b;\theta) \coloneqq \tfrac{1}{|b|^2} \sum_{i\in b} \sum_{j\in b} l((x_i, y_i), (x_j, y_j); \theta)\,. \quad (3.1)$$

We are interested specific class of embedding functions where a global average pooling is used. Specifically, consider the composite function family $e = g \circ f$ such that g is pooling and f is feature computation. We assume a further structure over the functions g and f. The feature function f maps the input space $\mathcal{X}$ into $\mathbb{R}^{w\times h\times d}$ where w and h are spatial dimensions. Moreover, g performs averaging as;

$$g(f(x;\theta)) = \frac{1}{w \cdot h} \sum_{i\in[w\cdot h]} f_i\,, \quad (3.2)$$

where $[n]{=}1, \ldots, n$ and we let $f_i{\in}\mathbb{R}^d$ denote i^{th} spatial feature of $f(x;\theta)$ to simplify notation. In the rest of the paper, we generalize the pooling function g into a learnable form and propose an algorithm to learn it.

4. Method

Consider the pooling operation in (3.2), it is a simple averaging over pixel-level feature maps (f_i). As we discuss in Sec. 1, one explanation for the effectiveness of this operation is considering each f_i as corresponding to a different semantic entity corresponding to the spatial extend of the pixel, and the averaging as convex combination over these semantic classes. Our method is based on generalizing this averaging such that a specific subset of pixels (correspondingly subset of semantic entities) are selected and their weights are adjusted according to their importance.

We generalize the pooling (3.2) in Sec. 4.1 by formulating a feature selection problem in which we prioritize a subset of the features that are closest to some trainable prototypes. If a feature is to be selected, its weight will be high. We then formulate our pooling operation as a differentiable layer to facilitate the prototype learning along with the rest of the embedding function parameters in Sec. 4.2. We learn the prototypes with class-level supervision, however in metric learning, learned representations should generalize to unseen classes. Thus, we introduce a zero-shot prediction loss to regularize prototype training for zero-shot setting in Sec. 4.3.

4.1. Generalized Sum Pooling as a Linear Program

Consider the pooling function g with adjustable weights as $g(f(x;\theta);\omega) = \sum_{i\in[n]} p_i f_i$ where $n = w\,h$. Note that, $p_i{=}1/n$ corresponds to average pooling. Informally, we want to control the weights to ease the metric learning problem. Specifically, we want the weights corresponding to background classes to be 0 and the ones corresponding to discriminative features to be high.

If we were given representations of discriminative semantic entities, we could simply compare them with the features (f_i) and choose the ones with high similarity. Our proposed method is simply learning these representations and using them for weight computations. We first discuss the weight computation part before discussing learning the representations of prototypes.

Assume that there are m discriminative semantic entities which we call *prototypes* with latent representations $\omega = \{\omega_i\}_{i\in[m]}$ of appropriate dimensions (same as f_i). Since we know that not all features ($\{f_i\}_{i\in[n]}$) are relevant, we need to choose a subset of $\{f_i\}_{i\in[n]}$. We perform this top-k selection process by converting it into an optimal transport (OT) problem.

Consider a cost map $c_{ij} = \|\bar{\omega}_i - \bar{f}_j\|_2$ which is an m (number of prototypes) by n (number of features)

matrix representing the closeness of prototypes ω_i and features f_j after some normalization $\bar{u} = u/\max\{1,\|u\|_2\}$. We aim to find a transport map π which re-distributes the uniform mass from features to prototypes. Since we do not have any prior information over features, we also consider its marginal distribution (importance of each feature to begin with) to be uniform. As we need to choose a subset, we set $\mu \in [0,1]$ ratio of mass to be transported. The resulting OT problem is:

$$\rho^*, \pi^* = \underset{\rho,\pi \geqslant 0}{\arg\min} \sum_{ij} c_{ij}\pi_{ij} \quad \text{s.to} \quad \rho_j + \underset{\sum_{ij}\pi_{ij}=\mu}{\Sigma_i \pi_{ij} = \frac{1}{n}} \ . \quad \text{(P1)}$$

Different to typical OT literature, we introduce decision variables ρ to represent residual weights to be discarded. Specifically modelling discarded weight instead of enforcing another marginalization constraint is beneficial beyond stylistic choices as it allows us to very efficiently compute gradients. When the introduced transport problem is solved, we perform weighting using residual weights as:

$$g(f(x;\theta);\omega) = \sum_i p_i f_i = \sum_i \frac{1/n - \rho_i^*}{\mu} f_i \,. \qquad (4.1)$$

Given set of prototypes $\{\omega_i\}_{i\in[m]}$, solving the problem in (P1) is a strict generalization of GAP since setting $\mu = 1$ recovers the original GAP. We formalize this equivalence in the following claim.

Theorem 4.1. *If $\mu = 1$, the operation in* (4.1) *reduces to global average pooling in* (3.2).

We defer the proof to Appendix. Having generalized GAP to a learnable form, we introduce a method to learn the prototypes $\{\omega_i\}_{i\in[m]}$ in the next section.

4.2. GSP as a Differentiable Layer

Consider the generalized form of pooling, defined as solution of (P1), as a layer of a neural network. The input is the feature vectors $\{f_i\}_{i\in[n]}$, the learnable parameters are prototype representations $\{\omega_i\}_{i\in[m]}$, and the output is residual weights ρ^*. To enable learning, we need partial derivatives of ρ^* with respect to $\{\omega_i\}_{i\in[m]}$. However, this function is not smooth. More importantly it requires the μ parameter to be known a priori.

We use a toy example to set the stage for rest of the formulation. Consider a 10×10×3 feature map visualized as RGB-image in Fig. 2 and corresponding two prototypes with representations $(1,0,0)$ (red) and $(0,0,1)$ (blue). The true $\mu = 0.5$ since the half of the image corresponds to red and blue, and other half is background class of green. Consider an under-estimation of $\mu = 0.2$, the global solution (shown as linear programming) is explicitly ignoring informative pixels (part of red and blue region). To solve this issue, we use entropy smoothing which is first introduced in [5] to enable fast computation of OT. Consider the entropy smoothed version of the original problem in (P1) as:

$$\rho^{(\varepsilon)}, \pi^{(\varepsilon)} = \underset{\substack{\rho,\pi\geqslant 0 \\ \rho_j + \Sigma_i \pi_{ij} = 1/n \\ \Sigma_{ij}\pi_{ij} = \mu}}{\arg\min} \sum_{ij} c_{ij}\pi_{ij} + \tfrac{1}{\varepsilon}(H(\pi) + H(\rho)), \quad \text{(P2)}$$

and obtain pooling weights by replacing ρ^* with $\rho^{(\varepsilon)}$ in (4.1), where $H(u) \coloneqq \Sigma_i u_i \log u_i$.

When smoothing is high ($\varepsilon \to 0$), the resulting solution is uniform over features similar to GAP. When it is low, the result is similar to top-k like behavior. For us, ε controls the trade-off between picking μ portion of the features that are closest to the prototypes and including as much features as possible for weight transfer. We further visualize the solution of the entropy smoothed problem in Fig. 2 showing desirable behavior even with underestimated μ.

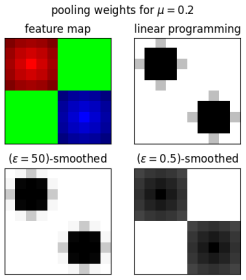

Figure 2: The resultant pooling weights (higher the darker) of different problems.

Beyond alleviating the under-estimation of μ problem, entropy smoothing also makes the problem strictly convex and smooth. Thus, the solution of the problem enables differentiation and in fact, admits closed-form gradient expression. We state the solution of (P2) and its corresponding gradient in the following propositions and defer their proofs to Appendix.

Proposition 4.2. *Given initialization $t^{(0)} = 1$, consider the following iteration:*

$$\rho^{(k+1)} = 1/n \,(1 + t^{(k)} \exp(-\varepsilon c)^\intercal \mathbf{1}_m)^{-1},$$
$$t^{(k+1)} = \mu \,(\mathbf{1}_m^\intercal \exp(-\varepsilon c)\rho^{(k+1)})^{-1}$$

where exp *and* $(\cdot)^{-1}$ *are element-wise and $\mathbf{1}_m$ is m-dimensional vector of ones. Then, $(\rho^{(k)}, t^{(k)})$ converges to the solution of* (P2) *defining transport map via $\pi^{(k)} = t^{(k)} \exp(-\varepsilon c) Diag(\rho^{(k)})$.*

Proposition 4.3. *Consider any differentiable loss function $\mathcal{L}$ as a function of (ρ, π). Given gradients $\frac{\partial \mathcal{L}}{\partial \rho}$ and $\frac{\partial \mathcal{L}}{\partial \pi}$, with (ρ, π) is the solution of* (P2). *Let $q = \rho \odot \frac{\partial \mathcal{L}}{\partial \rho} + (\pi \odot \frac{\partial \mathcal{L}}{\partial \pi})^\intercal \mathbf{1}_m$ and $\eta = (\rho \odot \frac{\partial \mathcal{L}}{\partial \rho})^\intercal \mathbf{1}_n - n\, q^\intercal \rho$, the gradient of $\mathcal{L}$ with respect to c reads:*

$$\frac{\partial \mathcal{L}}{\partial c} = -\varepsilon\Big(\pi \odot \frac{\partial \mathcal{L}}{\partial \pi} - n\pi Diag\big(q - \tfrac{\eta}{1 - \mu - n\rho^\intercal \rho}\big)\rho\Big) \ , \quad (4.2)$$

where $\odot$ denotes element-wise multiplication.

Proposition 4.2 and 4.3 suggest that our feature selective pooling can be implemented as a differentiable layer. Moreover, Proposition 4.3 gives a matrix inversion free computation of the gradient with respect to the costs unlike optimal transport based operators [36]. Thus, the prototypes, ω, can be jointly learned with the feature extraction efficiently.

4.3. Cross-batch Zero-shot Regularization

We formulated a prototype based feature pooling and learn the prototypes using class labels. Simply classifying the labels as prototypes is a degenerate solution. We rather want the prototypes to capture transferable attributes so that the learning can be transferred to the unseen classes as long as the attributes are shared. Learning with prototype based pooling shapes the embedding geometry in such a way that we have clusters corresponding to the prototypes in the embedding space. We want such clusters to have transferable semantics rather than class-specific information. To enable this, we now formulate a mechanism to predict class embedding vectors from the prototype assignment vectors and use that mechanism to tailor a loss regularizing the prototypes to have transferable representations.

Our feature selection layer should learn discriminative feature prototypes ω using top-down label information. Consider two randomly selected batches (b_1, b_2) of data sampled from the distribution. If the prototypes are corresponding to discriminative entities, the weights transferred to them (*i.e.*, marginal distribution of prototypes) should be useful in predicting the classes and such behavior should be consistent between batches for zero-shot prediction. Formally, if one class in b_2 does not exist in b_1, a predictor on class labels based on marginal distribution of prototypes for each class of b_1 should still be useful for b_2. DML losses do not carry such information. We thus formulate a zero-shot prediction loss to enforce such zero-shot transfer.

We consider that we are given an embedding vector υ_i for each class label i, *i.e.*, $\Upsilon = [\upsilon_i]_{i\in[c]}$ for c-many classes. We are to predict such embeddings from the marginal distribution of the prototypes. In particular, we use linear predictor A to predict label embeddings as $\hat{\upsilon} = A\,z$ where z is the normalized distribution of the weighs on the prototypes;

$$z = \tfrac{1}{\mu}\textstyle\sum_i \pi_i^{(\varepsilon)} \quad \text{where} \quad \pi^{(\varepsilon)} = [\pi_i^{(\varepsilon)}]_{i\in[n]}\,. \tag{4.3}$$

If we consider the prototypes as semantic vectors of some auxiliary labels such as *attributes* similar to zero-shot learning (ZSL) [7, 18, 62], then we can interpret z as *attribute* predictions. Given attribute predictions $\{z_i\}_{i\in b}$ and corresponding class embeddings for a batch b we fit the predictor as;

$$A_b = \underset{A=[a_i]_{i\in[m]}}{\arg\min} \textstyle\sum_{i\in b} \|A\,z_i - \upsilon_{y_i}\|_2^2 + \epsilon \sum_{i\in[m]} \|a_i\|_2^2\,, \tag{P3}$$

which admits a closed form expression enabling back propagation $A_b = \Upsilon_b\,(Z_b^\intercal Z_b + \epsilon I)^{-1} Z_b^\intercal$ where $\Upsilon_b = [\upsilon_{y_i}]_{i\in b}$, $Z_b = [z_i]_{i\in b}$. In practice, we are not provided with the label embeddings $\Upsilon = [\upsilon_i]_{i\in[c]}$. Nevertheless, having a closed-form expression for A_b enables us to exploit a meta-learning scheme like [3] to formulate a zero-shot prediction loss to learn them jointly with the rest of the parameters.

Specifically, we split a batch b into two[1] as b_1 and b_2 such that classes are disjoint. We then estimate attribute embeddings A_{b_k} according to (P3) using one set and use that estimate to predict the label embeddings of the other set to form zero-shot prediction loss. Formally, our loss becomes:

$$\begin{aligned}\mathcal{L}_{\text{ZS}}(b;\theta) = \tfrac{1}{|b_2|}\sum_{i\in b_2} \log\big(1 + \sum_{j\in[c]} \mathrm{e}^{(\upsilon_j - \upsilon_{y_i})^\intercal A_1\, z_i}\big) \\ + \tfrac{1}{|b_1|}\sum_{i\in b_1} \log\big(1 + \sum_{j\in[c]} \mathrm{e}^{(\upsilon_j - \upsilon_{y_i})^\intercal A_2\, z_i}\big),\end{aligned} \tag{4.4}$$

i.e., rearranged *soft-max cross-entropy* where $A_k{=}A_{b_k}$ with the abuse of notation, and $\theta = \{\theta_f, \omega, \Upsilon\}$ (*i.e.*, CNN parameters, prototype vectors, label embeddings).

We learn attribute embeddings (*i.e.*, columns of A) as sub-task and can define such learning as a differentiable operation. Intuitively, such a regularization should be useful in better generalization of our pooling operation to unseen classes since attribute predictions are connected to prototypes and the local features. We combine this loss with the metric learning loss using λ mixing (*i.e.*, (1-λ)$\mathcal{L}_{\text{DML}} + \lambda\mathcal{L}_{\text{ZS}}$) and jointly optimize.

4.4. Implementation Details

Embedding function. For the embedding function $f(\cdot;\theta)$ we use ResNet20 [17] for Cifar [30] experiments, and ImageNet [49] pretrained BN-Inception [19] for the rest. We exploit architectures until the output before the global average pooling layer. We add a per-pixel linear transform (*i.e.*, 1×1 convolution), to the output to obtain the local embedding vectors of size 128.

Pooling layer. For baseline methods, we use global average pooling. For our method, we perform parameter search and set the hyperparameters accordingly. Specifically, we use 64- or 128-many prototypes depending on the dataset. We use ε=0.5 for proxy-based losses and ε=5.0 for non-proxy losses. For the rest, we set μ=0.3, ϵ=0.05, λ=0.1 and we iterate until k=100 in Proposition 4.2. The embedding vectors upon pooling are $\ell 2$ normalized to have unit norm.

[1]Although we considered the simplest form which already worked well, repeating this splitting process can be beneficial.

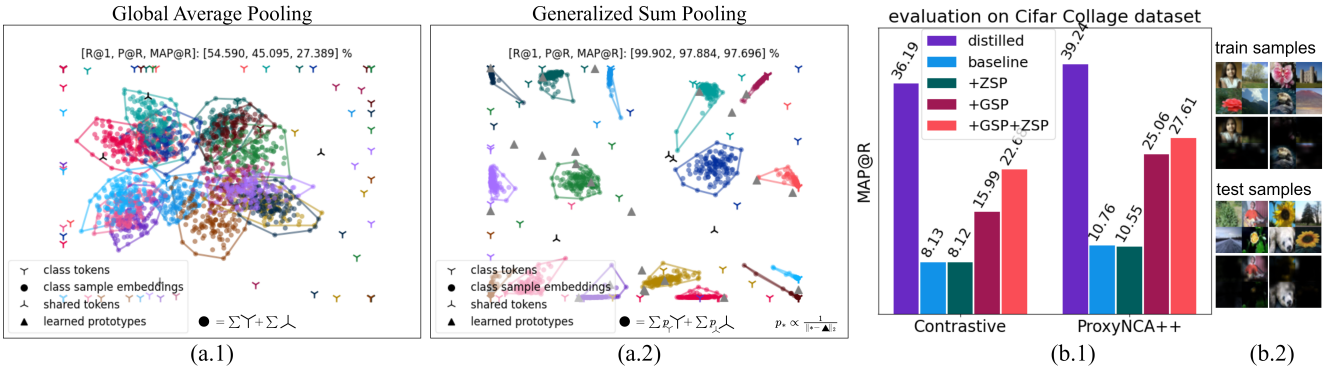

Figure 3: Behavior analysis of GSP: (a) GAP vs GSP in aggregating features. The tokens represent learned embedding vectors, and the samples are derived from their aggregation. (b) Experiments on Cifar Collage dataset: (b.1) presents the results, while (b.2) displays sample train and test images along with their attention maps in terms of pooling weights. *Distilled* denotes baseline performance on non-collage dataset, excluding the shared classes.

5. Experiments

We start our empirical study with a synthetic study validating the role of GAP in learning and the impact of GSP on the feature geometry. We further examine the effectiveness of our generalized sum pooling in metric learning for various models and datasets. We further perform ablation studies for the implications of our formulation as well as effects of the hyperparameters. We share the implementation details and the complete Tensorflow [1] code base in the supplemental materials.

5.1. Analysis of the Behavior

Synthetic study. We designed a synthetic empirical study to evaluate GSP in a fully controlled manner. We considered 16-class problem such that classes are defined over trainable tokens. In this setting, tokens correspond to semantic entities but we choose to give them a specific working to emphasize that they are trained as part of the learning. Each class is defined with 4 distinct tokens and there are also 4 background tokens shared by all classes. For example, a *"car"* class would have tokens like *"tire"* and *"window"* as well as background tokens of *"tree"* and *"road"*. We sampled class representations from both class specific and background tokens according to a mixing ratio $\tilde{\mu} \sim \mathcal{N}(0.5, 0.1)$. Such a 50-many feature collection corresponds to a training sample (*i.e.*, we are mimicking CNN's output with trainable tokens). We then obtained global representations using GAP and GSP. We visualize the geometry of the embedding space in Fig. 3-(a) along with the DML performances. With GAP, we observe overlapping class convex hulls resulting in poor DML performance. However, GSP gives well separated class convex hulls, further validating that it learns to ignore background tokens.

Selective pooling. We further extended this synthetic study to image domain by considering the 20 *super-classes* of Cifar100 dataset [30] where each has 5 sub-classes. For each super-class, we split the sub-classes for train (2), validation (1), and test (2). We consider 4 super-classes as the shared classes and compose 4x4-stitched collage images for the rest 16 classes (see supplementary material for details). In particular, we sample an image from a class and then sample 3 images from shared classes (Fig. 4). We used ResNet20 backbone pretrained on Cifar100 classification task and followed the implementation explained in Sec. 4.4. We provide the evaluation results in Fig. 3-(b). GSP and the proposed zero shot loss effectively increase MAP@R. We also provide sample train and test images to showcase that our pooling can transfer well to unseen classes.

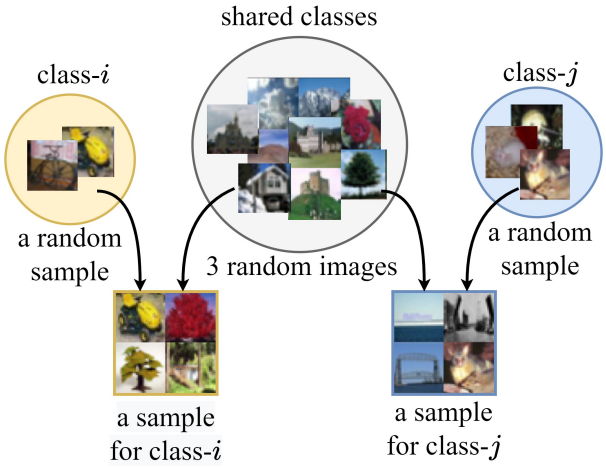

Figure 4: Sample generation for Cifar Collage dataset

Zero-shot regularization. We also evaluated the zero-shot prediction performance of the attribute vectors. We trained on Cifar10 dataset with 8 prototypes using ProxyNCA++ [53] (PNCA) loss with and without $\mathcal{L}_{\text{ZS}}$. We then extracted attribute histograms for each class and visualized them in Fig. 5. We observe transferable representations with $\mathcal{L}_{\text{ZS}}$ and we visually show in Fig. 5 that the semantic entities represented by the prototypes transfer across classes. We quantitatively evaluated such behavior by randomly splitting the classes into half and apply cross-batch zero-shot prediction explained in Sec. 4.3. Namely, we fit A in (P3) for one subset and used it to predict the class embeddings for the other set. We pre-computed class embeddings from the dataset as the class mean. To this end, our evaluation assesses generalization of both the features and the prototypes. We used MAP with both $\ell 2$ distance and *cosine* similarity in our evaluation. We repeated the experiment 1000 times. We observe in Fig. 5 that zero-shot performance of the prototypes learned with $\mathcal{L}_{\text{ZS}}$ is substantially superior. We also see that our feature aggregation method enables approximate localization of the semantic entities. Recent ZSL approaches [18, 62] can provide attribute localization and share a similar spirit with our method. However, attribute annotations must be provided for those methods whereas we exploit only class labels to extract attribute-like features. Thus, our method can be considered as an attribute-unsupervised alternative to them.

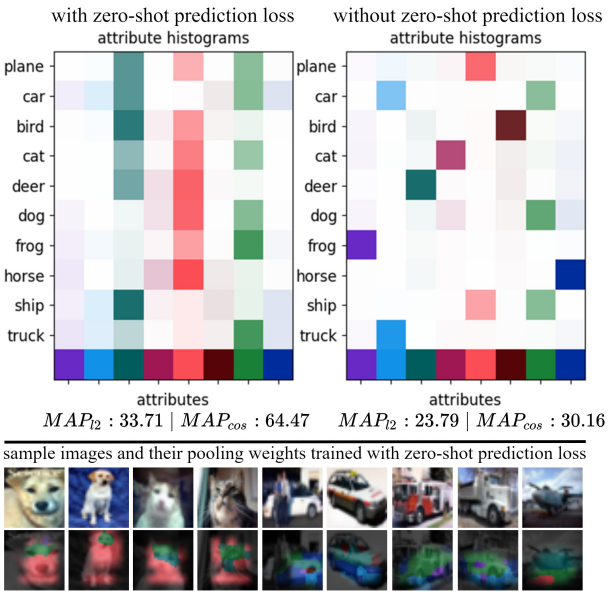

Figure 5: Comparing the distributions of the learned 8 prototypes across classes of Cifar10 dataset with and without $\mathcal{L}_{\text{ZS}}$. Pooling weights are coloured according to the dominant prototype at that location.

5.2. Deep Metric Learning Experiments

We largely rely on the recent work explicitly studying the fair evaluation strategies for metric learning [10, 39, 47]. Specifically, we follow the procedures proposed in [39] to evaluate our method as well as the other methods. We additionally follow the relatively old-fashioned conventional procedure [42] for the evaluation of our method and provide those results in the supplementary material. We provide full detail of our experimental setup in the supplementary material for complete transparency and reproducibility.

Datasets. CUB [56], Cars196 [29], In-shop [35], and SOP [42] with typical DML data augmentation [39].

Evaluation metrics. We report mean average precision (MAP@R) at R where R is defined for each query and is the total number of true references of the query.

Hyperparameters. We use Adam [26] optimizer with learning rate 10^{-5}, weight decay 10^{-4}, batch size 32 (4 per class). We train 4-fold: 4 models (1 for each $^3/_4$ train set partition).

Evaluation. Average performance (128D) where each of 4-fold model is trained 3 times resulting in realization of 3^4=81 different model collections. In our results we provide mean of 81 evaluations.

Baselines. We implement our method on top of and compare with *Contrastive (C2)*: Contrastive with positive margin [60], *MS*: Multi-similarity [57], *Triplet*: Triplet [51], *XBM*: Cross-batch memory [58] with contrastive loss [16], *PNCA*: ProxyNCA++ [53], *PAnchor*: ProxyAnchor [23].

5.2.1 Results

We compared our method (GSP) against direct application of GAP with 6 DML methods in 4 datasets. We also evaluated 14 additional pooling alternatives on *Ciffar Collage* and CUB datasets. We provide the tabulated results in supplementary material. Based on CUB performances, we picked generalized mean pooling (GeMean) [44] and DeLF [41] to compare against in 4 DML benchmarks. We also evaluated max pooling (GMP) and its combination with GAP as we typically observe GAP+GMP in the recent works [23, 53, 55, 58]. We also applied our method with GMP (GMP+GSP) and with GeMean (GeMean+GSP) to show that per channel selection is orthogonal to our method and thus, GSP can marginally improve those methods as well.

We present the tabulated evaluation results in Tab. 2, while Fig. 6 provides a concise summary of the relative MAP@R orderings of the methods employing 128D embeddings. We observe consistent improvements upon direct application of GAP in all datasets. On the average, we consistently improve the baselines ≈1% points

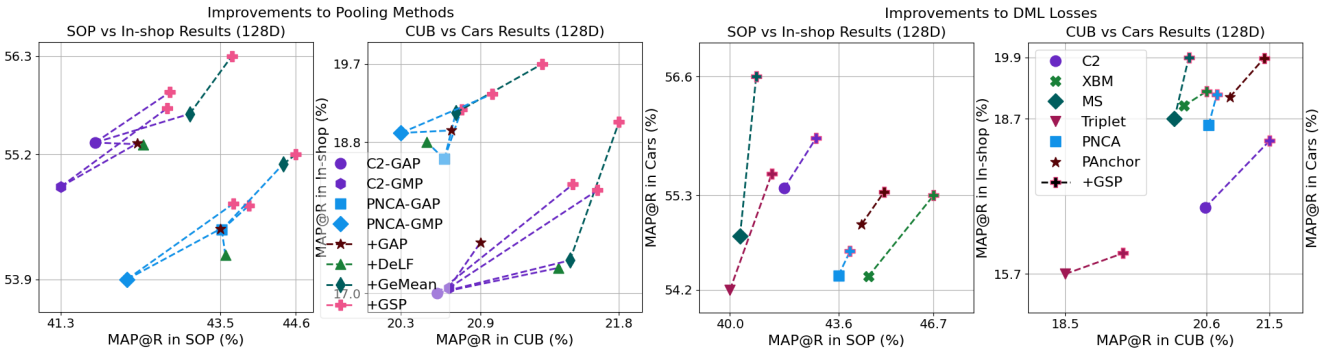

Figure 6: Summary of relative improvements: Each mark without a dashed line represents a baseline DML loss using GAP, unless stated otherwise. Similar losses are color-coded. Dashed lines indicate performance changes when replacing GAP with the associated pooling method (mark with the dashed line). Pooling methods applied on top of GMP or GMean are combined with them instead of replacing.

in MAP@R. Our improvement margins are superior to ones of attention based DeLF pooling. We improve state-of-the-art (SOTA) XBM method up to 2% points, which is a good evidence that application of GSP is not limited to loss terms but can be combined with different DML approaches. We also consistently improve GMP and GeMean pooling methods in all datasets, yet another evidence that our method can be combined with max pooling based methods.

We additionally evaluated our method with different architectures and methods in conventional setting [42] for the comparison with SOTA. The tabulated results are provided in supplementary material (§ 1.1), where we observe that we achieve SOTA performances with XBM [58] and LIBC [52] methods.

5.2.2 Ablations

Effect of $\mathcal{L}_{ZS}$. We empirically showed the effect of $\mathcal{L}_{ZS}$ on learned representations in Sec. 5.1. We further examined the effect of $\mathcal{L}_{ZS}$ quantitatively by enabling/disabling it. We also evaluated its effect without GSP by setting μ=1 where we used GAP with attribute vectors. The results are summarized in Tab. 1 showing that both components improve the baseline and their combination brings the best improvement. We observe similar behavior in *Cifar Collage* experiment (Fig. 3-(b)) where the effect of $\mathcal{L}_{ZS}$ is more substantial.

Table 1: Effects of $\mathcal{L}_{ZS}$ and GSP with C2 loss

		MAP@R							
		SOP		In-shop		CUB		Cars196	
$\mathcal{L}_{ZS}$	GSP	512D	128D	512D	128D	512D	128D	512D	128D
		45.85	41.79	59.07	55.38	25.95	20.58	24.38	17.02
	✓	46.78	42.66	59.46	55.50	26.25	20.85	25.54	17.88
✓		46.60	42.55	59.38	55.43	26.49	21.08	25.54	17.67
✓	✓	**46.81**	**42.84**	**60.01**	**55.94**	**27.12**	**21.52**	**26.25**	**18.31**

Computation efficiency. Through a series of k iterations, our pooling mechanism utilizes lightweight matrix-vector products to determine the pooling weights. While back propagation can be achieved through *automatic-differentiation*, it can become computationally intensive as k increases for certain problems. However, our pooling mechanism boasts a desirable feature of having a closed-form gradient expression for its backward computation, resulting in minimal scaling of computation load as k increases, as evidenced by our analysis in Fig. 7. We further provided the inference times for various k in supplementary material (§ 1.5).

Effect of μ. As outlined in Sec. 4.2, GSP is similar to top-k operator with an adaptive k thanks to entropy smoothing. We empirically validated such behavior by sweeping μ parameter controlling top-k behavior. The results, plotted in Fig. 7, show similar performance for lower μ values, with a decrease as μ increases, possibly due to overestimation of the foreground ratio. Hence a suggested value for μ is 0.3.

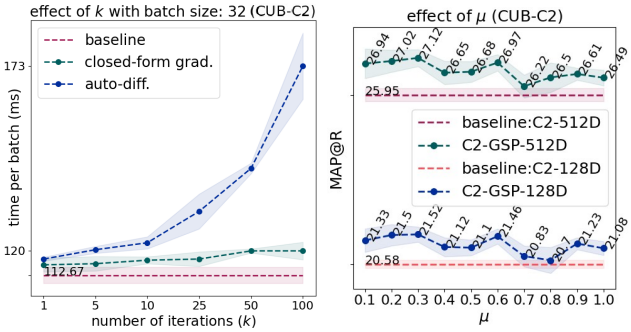

Figure 7: Efficiency of closed-form gradient (left) and effect of μ (right). Shaded regions represent $\mp std$.

6. Conclusion

We proposed a learnable and generalized version of GAP. Our proposed generalization GSP is a trainable pooling layer that selects the feature subset and re-weight it during pooling. To enable effective learning of our layer, we also proposed a cross-batch regularization improving zero-shot transfer. With extensive empirical studies, we validated the effectiveness of the proposed pooling layer in various DML benchmarks. We also established and empirically validated a valuable link between the computed transport maps for pooling and the prototypes, enabling attribute learning via meta-learning without explicit localized annotations. We believe such a connection is interesting, and has potential to offer pathways for enhanced embedding learning for DML as well as unsupervised attribute learning.

Table 2: Comparison with the existing methods for the retrieval task on SOP, In-shop, CUB, Cars. Experimental setting follows MLRC [39]. $\mp$ denotes 1 *std*. Red: the best, Blue: the second best, Bold: the loss term specific best.

Dataset →	**SOP**				**In-shop**				**CUB**				**Cars196**			
Dim. →	512D		128D		512D		128D		512D		128D		512D		128D	
Method↓	**P@1**	**MAP@R**	**P@1**	**MAP@R**	**P@1**	**MAP@R**	**P@1**	**MAP@R**	**P@1**	**MAP@R**	**P@1**	**MAP@R**	**P@1**	**MAP@R**	**P@1**	**MAP@R**
C1+																
GAP	69.29 ∓0.11	40.40 ∓0.15	65.15 ∓0.10	36.50 ∓0.11	80.11 ∓0.19	50.32 ∓0.14	75.83 ∓0.14	46.42 ∓0.13	63.32 ∓0.57	23.49 ∓0.31	56.34 ∓0.35	19.37 ∓0.29	78.01 ∓0.38	22.87 ∓0.33	65.61 ∓0.59	16.06 ∓0.14
XBM+GAP	76.54 ∓0.32	48.58 ∓0.47	73.22 ∓0.48	44.55 ∓0.57	87.76 ∓0.26	57.53 ∓0.41	85.26 ∓0.37	54.40 ∓0.45	65.56 ∓0.48	25.65 ∓0.24	57.48 ∓0.41	20.27 ∓0.19	83.55 ∓0.35	27.53 ∓0.22	72.17 ∓0.30	18.98 ∓0.17
XBM+GSP	**77.88** ∓0.18	**50.65** ∓0.28	**74.84** ∓0.19	**46.69** ∓0.28	**88.33** ∓0.19	**58.55** ∓0.29	**85.95** ∓0.21	**55.30** ∓0.21	**67.00** ∓0.49	**26.05** ∓0.15	**58.89** ∓0.49	**20.60** ∓0.16	**83.31** ∓0.22	**27.88** ∓0.23	**73.04** ∓0.39	**19.26** ∓0.19
C2+																
GAP	74.20 ∓0.23	45.85 ∓0.31	70.54 ∓0.19	41.79 ∓0.26	86.47 ∓0.15	59.07 ∓0.21	83.42 ∓0.12	55.38 ∓0.13	67.35 ∓0.50	25.95 ∓0.21	58.87 ∓0.36	20.58 ∓0.13	80.96 ∓0.48	24.38 ∓0.58	69.55 ∓0.42	17.02 ∓0.31
GSP	74.91 ∓0.12	46.81 ∓0.17	71.43 ∓0.11	42.84 ∓0.14	86.90 ∓0.17	**60.01** ∓0.29	83.57 ∓0.18	55.94 ∓0.17	**68.85** ∓0.41	27.12 ∓0.27	60.42 ∓0.36	21.52 ∓0.16	82.83 ∓0.27	26.25 ∓0.34	71.40 ∓0.27	18.31 ∓0.22
DeLF	74.59 ∓0.15	46.54 ∓0.19	45.53 ∓0.18	42.47 ∓0.17	86.65 ∓0.16	59.20 ∓0.22	83.51 ∓0.09	55.36 ∓0.12	68.66 ∓0.32	27.06 ∓0.18	59.85 ∓0.18	21.42 ∓0.16	81.85 ∓0.41	24.77 ∓0.38	69.95 ∓0.38	17.32 ∓0.25
GeMean	74.92 ∓0.13	46.99 ∓0.15	71.53 ∓0.11	43.12 ∓0.12	86.62 ∓0.15	59.12 ∓0.19	83.83 ∓0.09	55.70 ∓0.12	68.79 ∓0.36	27.12 ∓0.19	60.37 ∓0.30	21.50 ∓0.15	82.43 ∓0.60	25.27 ∓0.63	70.23 ∓0.55	17.41 ∓0.45
GeMean+GSP	**75.32** ∓0.08	**47.69** ∓0.13	**71.93** ∓0.10	**43.71** ∓0.13	**86.94** ∓0.15	59.98 ∓0.21	**84.35** ∓0.19	**56.34** ∓0.14	**69.11** ∓0.49	**27.56** ∓0.18	**60.81** ∓0.34	**21.84** ∓0.19	**83.62** ∓0.36	**26.98** ∓0.31	**72.38** ∓0.28	**19.05** ∓0.22
GMP	74.09 ∓0.15	46.13 ∓0.19	69.68 ∓0.20	41.31 ∓0.22	86.38 ∓0.12	59.04 ∓0.10	83.04 ∓0.13	54.89 ∓0.07	68.13 ∓0.40	26.43 ∓0.21	58.99 ∓0.34	20.66 ∓0.18	81.83 ∓0.62	25.11 ∓0.72	69.05 ∓0.61	17.08 ∓0.47
GMP+GAP	74.71 ∓0.11	46.70 ∓0.15	70.83 ∓0.10	42.38 ∓0.15	86.58 ∓0.16	59.22 ∓0.18	83.41 ∓0.12	55.37 ∓0.15	67.88 ∓0.48	26.63 ∓0.23	59.24 ∓0.32	20.88 ∓0.17	82.14 ∓0.40	25.66 ∓0.44	69.81 ∓0.38	17.62 ∓0.32
GMP+GSP	75.08 ∓0.1	47.12 ∓0.17	71.18 ∓0.15	42.80 ∓0.18	86.79 ∓0.16	59.43 ∓0.28	83.86 ∓0.15	55.76 ∓0.19	68.47 ∓0.58	**27.49** ∓0.36	60.19 ∓0.41	**21.69** ∓0.35	82.54 ∓0.46	26.30 ∓0.43	71.03 ∓0.48	18.24 ∓0.29
MS+																
GAP	72.81 ∓0.14	44.19 ∓0.21	69.09 ∓0.10	40.34 ∓0.16	87.01 ∓0.20	58.79 ∓0.37	83.87 ∓0.21	54.85 ∓0.34	65.43 ∓0.46	24.95 ∓0.15	**57.57** ∓0.27	20.13 ∓0.12	83.73 ∓0.34	27.16 ∓0.43	72.54 ∓0.43	18.73 ∓0.31
GSP	**73.05** ∓0.11	**44.72** ∓0.17	**69.44** ∓0.15	**40.87** ∓0.19	**88.28** ∓0.21	**60.49** ∓0.24	**85.28** ∓0.19	**56.62** ∓0.26	**65.50** ∓0.33	**25.09** ∓0.21	57.39 ∓0.15	**20.34** ∓0.22	**84.27** ∓0.35	**28.58** ∓0.40	**73.74** ∓0.32	**19.91** ∓0.31
Triplet+																
GAP	74.54 ∓0.24	45.88 ∓0.30	69.41 ∓0.38	40.01 ∓0.39	85.99 ∓0.36	59.67 ∓0.46	81.75 ∓0.38	54.25 ∓0.45	64.11 ∓0.66	23.65 ∓0.40	55.62 ∓0.46	18.54 ∓0.31	77.58 ∓0.60	22.67 ∓0.58	64.61 ∓0.59	15.74 ∓0.34
GSP	**75.59** ∓0.23	**47.35** ∓0.32	**70.65** ∓0.20	**41.38** ∓0.22	**86.75** ∓0.27	**60.85** ∓0.47	**82.74** ∓0.33	**55.54** ∓0.46	**66.09** ∓0.52	**24.80** ∓0.33	**57.12** ∓0.42	**19.38** ∓0.25	**78.93** ∓0.30	**23.44** ∓0.29	**65.81** ∓0.35	**16.14** ∓0.21
PNCA+																
GAP	75.18 ∓0.15	47.11 ∓0.16	72.15 ∓0.06	43.57 ∓0.08	87.26 ∓0.14	57.43 ∓0.14	84.86 ∓0.08	54.41 ∓0.10	65.74 ∓0.51	25.27 ∓0.23	58.19 ∓0.36	20.63 ∓0.20	82.33 ∓0.25	26.21 ∓0.22	70.75 ∓0.18	18.61 ∓0.08
GSP	75.68 ∓0.11	47.74 ∓0.14	72.37 ∓0.06	43.95 ∓0.06	87.35 ∓0.10	57.65 ∓0.12	85.13 ∓0.10	54.68 ∓0.08	65.80 ∓0.38	25.48 ∓0.25	58.20 ∓0.22	20.75 ∓0.19	82.70 ∓0.27	26.93 ∓0.18	71.55 ∓0.32	19.20 ∓0.17
DeLF	75.29 ∓0.09	47.44 ∓0.11	72.05 ∓0.06	43.62 ∓0.07	87.19 ∓0.11	57.44 ∓0.16	84.55 ∓0.04	54.13 ∓0.10	65.42 ∓0.34	25.31 ∓0.16	57.98 ∓0.24	20.51 ∓0.14	82.37 ∓0.35	26.63 ∓0.22	71.06 ∓0.27	18.81 ∓0.14
GeMean	75.64 ∓0.09	47.82 ∓0.09	72.75 ∓0.07	44.43 ∓0.06	87.63 ∓0.10	57.88 ∓0.13	85.48 ∓0.14	55.14 ∓0.12	66.33 ∓0.33	25.74 ∓0.20	58.52 ∓0.39	20.71 ∓0.20	**83.83** ∓0.29	27.44 ∓0.15	**72.14** ∓0.28	19.16 ∓0.12
GeMean+GSP	**75.89** ∓0.11	**48.17** ∓0.12	**72.91** ∓0.04	**44.61** ∓0.06	**87.64** ∓0.10	**58.12** ∓0.16	**85.58** ∓0.07	**55.25** ∓0.08	**67.39** ∓0.53	**26.19** ∓0.26	**59.39** ∓0.40	**21.31** ∓0.21	83.09 ∓0.25	**27.96** ∓0.30	71.95 ∓0.27	**19.74** ∓0.19
GMP	74.43 ∓0.08	46.33 ∓0.08	70.80 ∓0.07	42.24 ∓0.08	86.94 ∓0.13	56.79 ∓0.13	84.53 ∓0.08	53.86 ∓0.09	65.74 ∓0.51	25.36 ∓0.29	57.61 ∓0.38	20.33 ∓0.29	83.06 ∓0.33	26.96 ∓0.27	71.19 ∓0.25	18.92 ∓0.15
GMP+GAP	75.19 ∓0.09	47.26 ∓0.11	71.97 ∓0.04	43.55 ∓0.06	87.21 ∓0.14	57.34 ∓0.15	84.95 ∓0.09	54.42 ∓0.10	65.91 ∓0.35	25.56 ∓0.26	57.92 ∓0.37	20.68 ∓0.20	82.92 ∓0.41	26.92 ∓0.36	71.33 ∓0.22	18.95 ∓0.19
GMP+GSP	75.41 ∓0.12	47.50 ∓0.12	72.10 ∓0.07	43.73 ∓0.09	87.43 ∓0.10	57.68 ∓0.14	85.10 ∓0.10	54.70 ∓0.08	66.14 ∓0.48	25.85 ∓0.23	58.12 ∓0.32	20.96 ∓0.18	83.46 ∓0.31	27.12 ∓0.21	72.04 ∓0.39	19.38 ∓0.20
PAnchor+																
GAP	76.48 ∓0.19	48.08 ∓0.26	73.50 ∓0.14	44.33 ∓0.20	88.02 ∓0.21	58.02 ∓0.25	85.83 ∓0.18	54.98 ∓0.22	68.04 ∓0.41	26.20 ∓0.21	59.91 ∓0.34	20.94 ∓0.15	85.26 ∓0.31	27.14 ∓0.20	75.08 ∓0.23	19.15 ∓0.13
GSP	**77.13** ∓0.16	**49.05** ∓0.22	**74.07** ∓0.13	**45.07** ∓0.17	**88.10** ∓0.11	**58.44** ∓0.14	**85.97** ∓0.06	**55.34** ∓0.13	**68.40** ∓0.45	**26.59** ∓0.25	**60.80** ∓0.31	**21.44** ∓0.17	**86.46** ∓0.39	**28.43** ∓0.33	**75.88** ∓0.25	**19.90** ∓0.20

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[2] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5297–5307, 2016.

[3] Luca Bertinetto, Joao F Henriques, Philip Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations*, 2018.

[4] Oğul Can, Yeti Z Gürbüz, and A Aydın Alatan. Deep metric learning with alternating projections onto feasible sets. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 1264–1268. IEEE, 2021.

[5] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013.

[6] Marco Cuturi, Olivier Teboul, and Jean-Philippe Vert. Differentiable ranking and sorting using optimal transport. *Advances in neural information processing systems*, 32, 2019.

[7] Berkan Demirel, Ramazan Gokberk Cinbis, and Nazli Ikizler-Cinbis. Attributes2classname: A discriminative model for attribute-based unsupervised zero-shot learning. In *Proceedings of the IEEE international conference on computer vision*, pages 1232–1241, 2017.

[8] M Dong, X Yang, R Zhu, Y Wang, and J Xue. Generalization bound of gradient descent for non-convex metric learning. Neural Information Processing Systems Foundation, 2020.

[9] Aleksandr Ermolov, Leyla Mirvakhabova, Valentin Khrulkov, Nicu Sebe, and Ivan Oseledets. Hyperbolic vision transformers: Combining improvements in metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7409–7419, 2022.

[10] Istvan Fehervari, Avinash Ravichandran, and Srikar Appalaraju. Unbiased evaluation of deep metric learning algorithms. *arXiv preprint arXiv:1911.12528*, 2019.

[11] Zilin Gao, Jiangtao Xie, Qilong Wang, and Peihua Li. Global second-order pooling convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3024–3033, 2019.

[12] Geonmo Gu, Byungsoo Ko, and Han-Gyu Kim. Proxy synthesis: Learning with synthetic classes for deep metric learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 1460–1468, 2021.

[13] Yeti Z Gürbüz and A Aydın Alatan. A novel bovw mimicking end-to-end trainable cnn classification framework using optimal transport theory. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 3053–3057. IEEE, 2019.

[14] Yeti Z Gurbuz and A Aydin Alatan. Generalizable embeddings with cross-batch metric learning. *arXiv preprint arXiv:2307.07620*, 2023.

[15] Yeti Z Gurbuz, Ogul Can, and A Aydin Alatan. Deep metric learning with chance constraints. *arXiv preprint arXiv:2209.09060*, 2022.

[16] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[18] Dat Huynh and Ehsan Elhamifar. Fine-grained generalized zero-shot learning via dense attribute-based attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4483–4493, 2020.

[19] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

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

[21] Yannis Kalantidis, Clayton Mellina, and Simon Osindero. Cross-dimensional weighting for aggregated deep convolutional features. In *European conference on computer vision*, pages 685–701. Springer, 2016.

[22] SHICHAO KAN, Yixiong Liang, Min Li, Yigang Cen, Jianxin Wang, and Zhihai He. Coded residual transform for generalizable deep metric learning. In *Advances in Neural Information Processing Systems*.

[23] Sungyeon Kim, Dongwon Kim, Minsu Cho, and Suha Kwak. Proxy anchor loss for deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3238–3247, 2020.

[24] Wonsik Kim, Bhavya Goyal, Kunal Chawla, Jungmin Lee, and Keunjoo Kwon. Attention-based ensemble for deep metric learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 736–751, 2018.

[25] Yonghyun Kim and Wonpyo Park. Multi-level distance regularization for deep metric learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 1827–1835, 2021.

[26] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[27] Byungsoo Ko and Geonmo Gu. Embedding expansion: Augmentation in embedding space for deep metric

learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7255–7264, 2020.

[28] Soheil Kolouri, Navid Naderializadeh, Gustavo K Rohde, and Heiko Hoffmann. Wasserstein embedding for graph learning. In *International Conference on Learning Representations*, 2020.

[29] Andreas Krause and Daniel Golovin. Submodular function maximization. In *Tractability: Practical Approaches to Hard Problems*, pages 71–104. Cambridge University Press, 2014.

[30] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[31] Yunwen Lei, Mingrui Liu, and Yiming Ying. Generalization guarantee of sgd for pairwise learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[32] Jongin Lim, Sangdoo Yun, Seulki Park, and Jin Young Choi. Hypergraph-induced semantic tuplet loss for deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 212–222, 2022.

[33] Xudong Lin, Yueqi Duan, Qiyuan Dong, Jiwen Lu, and Jie Zhou. Deep variational metric learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 689–704, 2018.

[34] Chang Liu, Han Yu, Boyang Li, Zhiqi Shen, Zhanning Gao, Peiran Ren, Xuansong Xie, Lizhen Cui, and Chunyan Miao. Noise-resistant deep metric learning with ranking-based instance selection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6811–6820, 2021.

[35] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1096–1104, 2016.

[36] Giulia Luise, Alessandro Rudi, Massimiliano Pontil, and Carlo Ciliberto. Differential properties of sinkhorn approximation for learning with wasserstein distance. *Advances in Neural Information Processing Systems*, 31, 2018.

[37] Grégoire Mialon, Dexiong Chen, Alexandre d'Aspremont, and Julien Mairal. A trainable optimal transport embedding for feature aggregation and its relationship to attention. In *ICLR 2021-The Ninth International Conference on Learning Representations*, 2021.

[38] Timo Milbich, Karsten Roth, Homanga Bharadhwaj, Samarth Sinha, Yoshua Bengio, Björn Ommer, and Joseph Paul Cohen. Diva: Diverse visual feature aggregation for deep metric learning. In *European Conference on Computer Vision*, pages 590–607. Springer, 2020.

[39] Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. A metric learning reality check. In *European Conference on Computer Vision*, pages 681–699. Springer, 2020.

[40] Navid Naderializadeh, Joseph F Comer, Reed Andrews, Heiko Hoffmann, and Soheil Kolouri. Pooling by sliced-wasserstein embedding. *Advances in Neural Information Processing Systems*, 34:3389–3400, 2021.

[41] Hyeonwoo Noh, Andre Araujo, Jack Sim, Tobias Weyand, and Bohyung Han. Large-scale image retrieval with attentive deep local features. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[42] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4004–4012, 2016.

[43] Yash Patel, Giorgos Tolias, and Jiří Matas. Recall@ k surrogate loss with large batches and similarity mixup. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7502–7511, 2022.

[44] Filip Radenović, Giorgos Tolias, and Ondřej Chum. Fine-tuning cnn image retrieval with no human annotation. *IEEE transactions on pattern analysis and machine intelligence*, 41(7):1655–1668, 2018.

[45] Karsten Roth, Biagio Brattoli, and Bjorn Ommer. Mic: Mining interclass characteristics for improved metric learning. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[46] Karsten Roth, Timo Milbich, Bjorn Ommer, Joseph Paul Cohen, and Marzyeh Ghassemi. S2sd: Simultaneous similarity-based self-distillation for deep metric learning. In *ICML 2021: 38th International Conference on Machine Learning*, pages 9095–9106, 2021.

[47] Karsten Roth, Timo Milbich, Samarth Sinha, Prateek Gupta, Bjorn Ommer, and Joseph Paul Cohen. Revisiting training strategies and generalization performance in deep metric learning. In *International Conference on Machine Learning*, pages 8242–8252. PMLR, 2020.

[48] Karsten Roth, Oriol Vinyals, and Zeynep Akata. Non-isotropy regularization for proxy-based deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7420–7430, 2022.

[49] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[50] Artsiom Sanakoyeu, Vadim Tschernezki, Uta Buchler, and Bjorn Ommer. Divide and conquer the embedding space for metric learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[51] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE confer-*

ence on computer vision and pattern recognition, pages 815–823, 2015.

[52] Jenny Seidenschwarz, Ismail Elezi, and Laura Leal-Taixé. Learning intra-batch connections for deep metric learning. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 9410–9421. PMLR, 2021.

[53] Eu Wern Teh, Terrance DeVries, and Graham W Taylor. Proxynca++: Revisiting and revitalizing proxy neighborhood component analysis. In *European Conference on Computer Vision (ECCV)*. Springer, 2020.

[54] Giorgos Tolias, Tomas Jenicek, and Ondřej Chum. Learning and aggregating deep local descriptors for instance-level recognition. In *European Conference on Computer Vision*, pages 460–477. Springer, 2020.

[55] Shashanka Venkataramanan, Bill Psomas, Ewa Kijak, laurent amsaleg, Konstantinos Karantzalos, and Yannis Avrithis. It takes two to tango: Mixup for deep metric learning. In *International Conference on Learning Representations*, 2022.

[56] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[57] Xun Wang, Xintong Han, Weilin Huang, Dengke Dong, and Matthew R. Scott. Multi-similarity loss with general pair weighting for deep metric learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[58] Xun Wang, Haozhi Zhang, Weilin Huang, and Matthew R Scott. Cross-batch memory for embedding learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6388–6397, 2020.

[59] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[60] Chao-Yuan Wu, R Manmatha, Alexander J Smola, and Philipp Krahenbuhl. Sampling matters in deep embedding learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2840–2848, 2017.

[61] Yujia Xie, Hanjun Dai, Minshuo Chen, Bo Dai, Tuo Zhao, Hongyuan Zha, Wei Wei, and Tomas Pfister. Differentiable top-k with optimal transport. *Advances in Neural Information Processing Systems*, 33:20520–20531, 2020.

[62] Wenjia Xu, Yongqin Xian, Jiuniu Wang, Bernt Schiele, and Zeynep Akata. Attribute prototype network for zero-shot learning. *Advances in Neural Information Processing Systems*, 33:21969–21980, 2020.

[63] Hong Xuan, Richard Souvenir, and Robert Pless. Deep randomized ensembles for metric learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 723–734, 2018.

[64] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[65] Dingyi Zhang, Yingming Li, and Zhongfei Zhang. Deep metric learning with spherical embedding. *Advances in Neural Information Processing Systems*, 33, 2020.

[66] Hang Zhang, Jia Xue, and Kristin Dana. Deep ten: Texture encoding network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 708–717, 2017.

[67] Wenliang Zhao, Yongming Rao, Ziyi Wang, Jiwen Lu, and Jie Zhou. Towards interpretable deep metric learning with structural matching. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9887–9896, 2021.

[68] Wenzhao Zheng, Chengkun Wang, Jiwen Lu, and Jie Zhou. Deep compositional metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9320–9329, 2021.

[69] Wenzhao Zheng, Borui Zhang, Jiwen Lu, and Jie Zhou. Deep relational metric learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12065–12074, 2021.

[70] Bolei Zhou, David Bau, Aude Oliva, and Antonio Torralba. Interpreting deep visual representations via network dissection. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[71] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.