Robust and Decomposable Average Precision for Image Retrieval

Elias Ramzi, Nicolas Thome, Clément Rambour, Nicolas Audebert, Xavier Bitot

Robust and Decomposable Average Precision for Image Retrieval

Elias Ramzi[1,2]
elias.ramzi@cnam.fr

Nicolas Thome[1]
nicolas.thome@cnam.fr

Clément Rambour[1]
clement.rambour@cnam.fr

Nicolas Audebert[1]
nicolas.audebert@cnam.fr

Xavier Bitot[2]
xavier.bitot@coexya.eu

[1]CEDRIC, Conservatoire National des Arts et Métiers, Paris, France
[2]Coexya, Paris, France

Abstract

In image retrieval, standard evaluation metrics rely on score ranking, e.g. average precision (AP). In this paper, we introduce a method for robust and decomposable average precision (ROADMAP) addressing two major challenges for end-to-end training of deep neural networks with AP: non-differentiability and non-decomposability. Firstly, we propose a new differentiable approximation of the rank function, which provides an upper bound of the AP loss and ensures robust training. Secondly, we design a simple yet effective loss function to reduce the decomposability gap between the AP in the whole training set and its averaged batch approximation, for which we provide theoretical guarantees. Extensive experiments conducted on three image retrieval datasets show that ROADMAP outperforms several recent AP approximation methods and highlight the importance of our two contributions. Finally, using ROADMAP for training deep models yields very good performances, outperforming state-of-the-art results on the three datasets. Code and instructions to reproduce our results will be made publicly available at `https://github.com/elias-ramzi/ROADMAP`.

1 Introduction

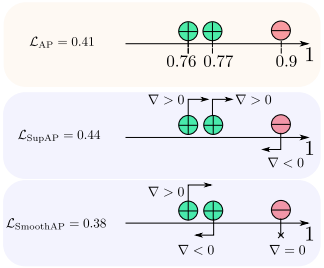
The task of 'query by example' is a major prediction problem, which consists in learning a similarity function able to properly rank all the instances in a retrieval set according to their relevance to the query, such that relevant items have the largest similarity. In computer vision, it drives several major applications, *e.g.* content-based image retrieval, face recognition or person re-identification.

Such tasks are usually evaluated with rank-based metrics, *e.g.* Recall@k, Normalized Discounted Cumulative Gain (NDCG), and Average Precision (AP). AP is also the *de facto* metric used in several vision tasks implying a large imbalance between positive and negative samples, *e.g.* object detection.

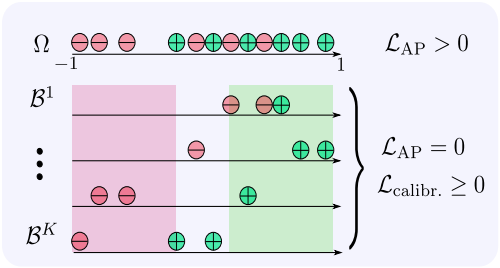
In this paper, we address the problem of direct AP training with stochastic gradient-based optimization, *e.g.* using deep neural networks, which poses two major challenges.

Firstly, the AP loss $\mathcal{L}_{\text{AP}} = 1 - \text{AP}$ is not differentiable and is thus not directly amenable to gradient-based optimization. There has been a rich literature for providing smooth and upper bound surrogate

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.



(a) $\mathcal{L}_{\text{SupAP}} \geq \mathcal{L}_{\text{AP}}$ and $\nabla \mathcal{L}_{\text{SupAP}} > 0$ in this example, in contrast to SmoothAP [2]. This ensures robust training and comes from a new approximation of the rank function.



(b) $\mathcal{L}_{\text{AP}}$ non-decomposability: $\mathcal{L}_{\text{AP}} = 0$ in all batches $\mathcal{B}^i$ despite $\mathcal{L}_{\text{AP}} \neq 0$ over the whole $\bigcup_i \mathcal{B}^i$. $\mathcal{L}_{\text{calibr.}}$ controls the absolute scores between batches, such that $\mathcal{L}_{\text{ROADMAP}} \neq 0$ in each batch.

Figure 1: Our robust and decomposable Average Precision training (ROADMAP) includes (a) a smooth loss $\mathcal{L}_{\text{SupAP}}$ upper-bounding $\mathcal{L}_{\text{AP}}$, and (b) a calibration loss $\mathcal{L}_{\text{calibr.}}$ supporting decomposability.

losses for $\mathcal{L}_{\text{AP}}$ [50, 23, 24, 6, 28]. More recently, smooth differentiable rank approximations have been proposed [40, 15, 16, 3, 32, 8, 2], but generally lose the important $\mathcal{L}_{\text{AP}}$ upper bound property.

The second important issue of AP optimization relates to its non-decomposability: $\mathcal{L}_{\text{AP}}^B$ averaged over batches underestimates $\mathcal{L}_{\text{AP}}$ on the whole training dataset, which we refer as the *decomposability gap*. In image retrieval, the attempts to circumvent the problem involve *ad hoc* methods based on batch sampling strategies [10, 37, 22, 37, 35], or storing all training representations/scores [44, 3, 32, 28], leading to complex models with a large computation and memory overhead.

In this paper, we introduce a method for RObust And DecoMposable Average Precision (ROADMAP), which explicitly addresses the aforementioned challenges of AP optimization.

Our first contribution is to propose a new surrogate loss $\mathcal{L}_{\text{SupAP}}$ for $\mathcal{L}_{\text{AP}}$. In particular, we introduce a smooth approximation of the rank function, with a different behaviour for positive and negative examples. By this design, $\mathcal{L}_{\text{SupAP}}$ provides an upper bound of $\mathcal{L}_{\text{AP}}$, and always back-propagates gradients when the correct ranking is not satisfied. These two features illustrated in the the toy example on Figure are not fulfilled by binning approaches [3, 32] or by SmoothAP [2].

As a second contribution, we propose to improve the non-decomposability in AP training. To this end, we introduce a simple yet effective training objective $\mathcal{L}_{\text{calibr.}}$, which calibrates the scores among different batches by controlling the absolute value of positive and negative samples. We provide a theoretical analysis showing that $\mathcal{L}_{\text{calibr.}}$ decreases the decomposability gap. Figure 1b illustrates how $\mathcal{L}_{\text{calibr.}}$ can be leveraged to improve the overall ranking.

We provide a thorough experimental validation including three standard image retrieval datasets and show that ROADMAP outperforms state-of-the-art methods. We also report the large and consistent gain compared to rank/AP approximation baselines, and we highlight in the ablation studies the importance of our two contributions. Finally, ROADMAP does not entail any memory or computation overhead and remains competitive even with small batches.

2 Related work

We discuss here the literature in image retrieval dedicated to AP optimization, and compare to other approaches based on optimizing representations [25, 1, 51, 53, 38] in the experiments.

Smooth AP approximations Studying smooth surrogate losses for AP has a long history. The widely used surrogate for retrieval is to consider constraints based on pairs [47, 12, 31], triplets [11], quadruplets [20] or n-uplets [35] to enforce partial ranking. These metric learning methods optimize a very coarse upper bound on AP and need complex post-processing and tricks to be effective.

One option for training with AP is to design smooth upper bounds on the AP loss. Seminal works are based on structural SVMs [50, 23], with extensions to speed-up the "loss-augmented inference" [24] or to adapt to weak supervision [6]. Recently, a generic blackbox combinatorial solver has been introduced [28] and applied to AP optimization [33]. To overcome the brittleness of AP with respect to

small score variations, an *ad hoc* perturbation is applied to positive and negative scores during training. These methods provide elegant AP upper bounds, but generally are coarse AP approximations.

Other approaches rely on designing smooth approximations of the the rank function. This is done in soft-binning techniques [15, 16, 40, 3, 32] by using a smoothed discretization of similarity scores. Other approaches rely on explicitly approximating the non-differentiable rank functions using neural networks [8], or with a sum of sigmoid functions in the recent SmoothAP approach [2]. These approaches enable accurate AP approximations by providing tight and smooth approximations of the rank function. However, they do not guarantee that the resulting loss is an AP loss upper bound. The $\mathcal{L}_{\text{SupAP}}$ introduced in this work is based on a smooth approximation of the rank function leading to an upper bound on the AP loss, making our approach both accurate and robust.

Decomposability in AP optimization Batch training is mandatory in deep learning. However, the non-decomposability of AP is a severe issue, since it yields an inconsistent AP gradient estimator.

Non-decomposability is related to sampling informative constraints in simple AP surrogates, *e.g.* triplet losses, since the constraints' cardinality on the whole training set is prohibitive. This has been addressed by efficient batch sampling [13, 10, 37] or selecting informative constraints within mini-batches [35, 9, 4, 37]. In cross-batch memory technique [44], the authors assume a slow drift in learned representations to store them and compute global mining in pair-based deep metric learning.

In AP optimization, the non-decomposability has essentially been addressed by a brute force increase of the batch size [3, 32, 28]. This includes an important overhead in computation and memory, generally involving a two-step approach for first computing the AP loss and subsequently re-computing activations and back-propagating gradients. In contrast, our loss $\mathcal{L}_{\text{calibr.}}$ does not add any overhead and enables good performances for AP optimization even with small batches.

3 Robust and decomposable AP training

We present here our method for RObust And DecoMposable AP (ROADMAP) dedicated to direct optimization of a smooth surrogate of AP with stochastic gradient descent (SGD), see Fig. 2.

Training context Let us consider a retrieval set $\Omega = \{\boldsymbol{x_j}\}_{j\in[\![1;N]\!]}$ composed of N elements, and a set of M queries included in Ω, *i.e.* $\mathcal{Q} = \{\boldsymbol{q_i}\}_{i\in[\![1;M]\!]} \subseteq \Omega$. For each query $\boldsymbol{q_i}$, each element in Ω is assigned a label $y(\boldsymbol{x_j}, \boldsymbol{q_i}) \in \{+1; -1\}$, such that $y(\boldsymbol{x_j}, \boldsymbol{q_i}) = 1$ (resp. $y(\boldsymbol{x_j}, \boldsymbol{q_i}) = -1$) if $\boldsymbol{x_j}$ is relevant (resp. irrelevant) with respect to $\boldsymbol{q_i}$. This defines a query-dependent partitioning of Ω such that $\Omega = \mathcal{P}_i \cup \mathcal{N}_i$, where $\mathcal{P}_i := \{\boldsymbol{x_j} \in \Omega | y(\boldsymbol{x_j}, \boldsymbol{q_i}) = +1\}$ and $\mathcal{N}_i := \{\boldsymbol{x_j} \in \Omega | y(\boldsymbol{x_j}, \boldsymbol{q_i}) = -1\}$.

For each $\boldsymbol{x_j} \in \Omega$, we define a prediction model parametrized by parameters $\boldsymbol{\theta}$, *e.g.* a deep neural network, which provides a vectorial embedding $\mathbf{v}_{\boldsymbol{q_i}} \in \mathbb{R}^d$ of each element, *i.e.*: $\mathbf{v}_{\boldsymbol{q_i}} := f_{\boldsymbol{\theta}}(\boldsymbol{q_i})$. In the embedded space $\mathbb{R}^d$, we compute a similarity score between each query $\boldsymbol{q_i}$ and each element in Ω, *e.g.* by using the cosine similarity: $s(\boldsymbol{q_i}, \boldsymbol{x_j}) = \frac{\mathbf{v}_{\mathbf{q_i}}{}^T \mathbf{v}_{\mathbf{j}}}{||\mathbf{v}_{\mathbf{q_i}}||^2 ||\mathbf{v}_{\mathbf{j}}||^2}$.

During training, our goal is to optimize, for each query $\boldsymbol{q_i}$, the model parameters $\boldsymbol{\theta}$ such that positive elements are ranked before negatives. More precisely, we aim at minimizing the AP loss $\mathcal{L}_{\text{AP}_i}$ for each query $\boldsymbol{q_i}$ in the retrieval set Ω.

Our overall AP loss $\mathcal{L}_{\text{AP}}$ is averaged over all queries:

$$\mathcal{L}_{\text{AP}}(\boldsymbol{\theta}) = 1 - \frac{1}{M}\sum_{i=1}^{M} \text{AP}_i(\boldsymbol{\theta}), \quad \text{AP}_i(\boldsymbol{\theta}) = \frac{1}{|\mathcal{P}_i|}\sum_{k\in\mathcal{P}_i} \text{Pre}(k,\theta) = \frac{1}{|\mathcal{P}_i|}\sum_{k\in\mathcal{P}_i} \frac{\text{rank}^+(k,\theta)}{\text{rank}(k,\theta)} \tag{1}$$

where $\text{Pre}(k,\theta)$ is the precision for the k^{th} positive example $\boldsymbol{x_k}$, $\text{rank}^+(k,\theta)$ its rank among positives $\mathcal{P}_i$, and the $\text{rank}(k,\theta)$ its rank over $\Omega = \mathcal{P}_i \cup \mathcal{N}_i$.

As previously mentioned, there are two main challenges with SGD optimization of AP in Eq. (1): i) $\text{AP}(\boldsymbol{\theta})$ is not differentiable with respect to $\boldsymbol{\theta}$, and ii) AP does not linearly decompose into batches. ROADMAP addresses both issues: we introduce the robust differentiable $\mathcal{L}_{\text{SupAP}}$ surrogate (Section 3.1), and add the $\mathcal{L}_{\text{calibr.}}$ loss (Section 3.2) to improve AP decomposability. Our final loss $\mathcal{L}_{\text{ROADMAP}}$ is a linear combination of $\mathcal{L}_{\text{SupAP}}$ and $\mathcal{L}_{\text{calibr.}}$, weighted by the hyperparameter λ:

$$\mathcal{L}_{\text{ROADMAP}}(\boldsymbol{\theta}) = (1-\lambda)\cdot\mathcal{L}_{\text{SupAP}}(\boldsymbol{\theta}) + \lambda\cdot\mathcal{L}_{\text{calibr.}}(\boldsymbol{\theta}) \tag{2}$$

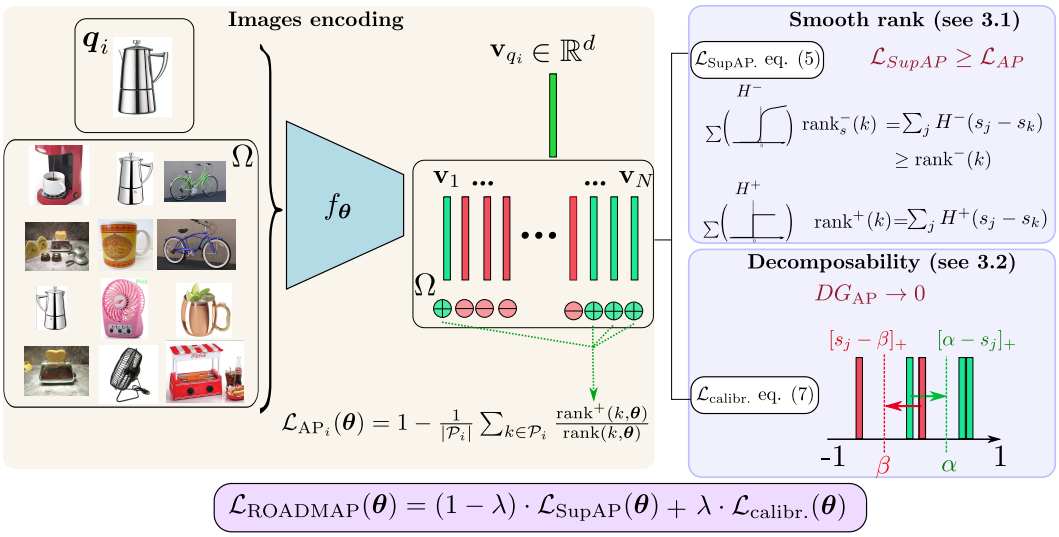


Figure 2: ROADMAP training: we optimize parameters $\boldsymbol{\theta}$ of a deep neural networks to minimize a smooth surrogate of $\mathcal{L}_{\mathrm{AP}_i}(\boldsymbol{\theta})$ between the query $\boldsymbol{q_i}$ and the retrieval set Ω. Our smooth rank approximations H^+ and H^- enables $\mathcal{L}_{\mathrm{SupAP}}$ to be both accurate and robust (sec 3.1), and $\mathcal{L}_{\mathrm{calibr.}}$ enables an implicit batch scores comparison for better decomposability without additional storing (sec 3.2).

3.1 Robustness in smooth rank approximation

The non-differentiablity in Eq (1) comes from the ranking operator, which can be viewed as counting the number of instances that have a similarity score greater than the considered instance, *i.e.*[1]:

$$\mathrm{rank}^+(k) = 1 + \sum_{j \in \mathcal{P}_i \setminus \{k\}} H(s_j - s_k), \quad \text{where } H(t) = \begin{cases} 1 & \text{if } t \geq 0 \\ 0 & \text{otherwise} \end{cases}$$
$$\mathrm{rank}(k) = \mathrm{rank}^+(k) + \sum_{j \in \mathcal{N}_i} H(s_j - s_k) = \mathrm{rank}^+(k) + \mathrm{rank}^-(k) \tag{3}$$

From Eq. (3) it becomes clear that the non-differentiablity is due to the Heaviside (step) function H, whose derivative is either zero or undefined. Note that the computation of $\mathrm{rank}^+(k)$ and $\mathrm{rank}^-(k)$ in Eq. (3) relates to the rank of positive instances $\boldsymbol{x_k} \in \mathcal{P}_i$: the score s_k in Eq. (3) is always the score of a positive, whereas s_j can either be a negative's or positive's score.

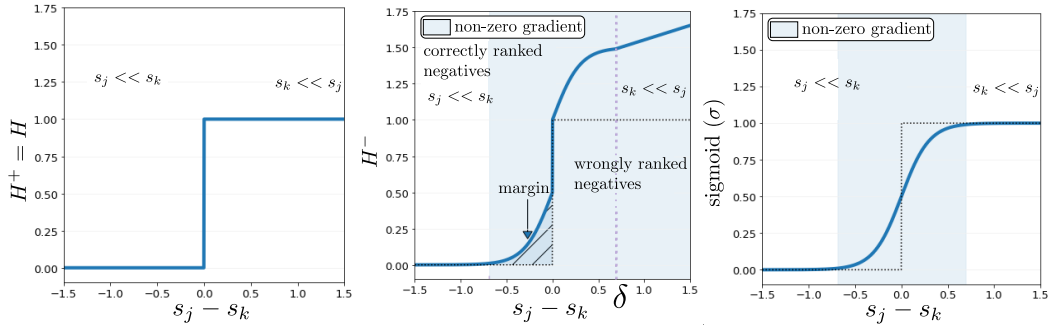
Smooth loss $\mathcal{L}_{\mathrm{SupAP}}$ To provide a smooth approximation of $\mathcal{L}_{\mathrm{AP}}$ in Eq. (1), we introduce a smooth approximation of the rank function. In particular, we propose a different behaviour between $\mathrm{rank}^+(k)$ and $\mathrm{rank}^-(k)$ in Eq. (3) by defining two functions H^+ and H^-.

For $\mathrm{rank}^+(k)$, we choose to keep the Heaviside (step) function, *i.e.* $H^+ = H$ (see Fig. 3a), which consists in ignoring $\mathrm{rank}^+(k)$ in gradient-based AP optimization. This is done on purpose since $\frac{\partial AP}{\partial\, \mathrm{rank}^+(k)} = \frac{\mathrm{rank}^-(k)}{(\mathrm{rank}^+(k)+\mathrm{rank}^-(k))^2} \geq 0$: the gradient would tend to increase $\mathrm{rank}^+(k)$ and to decrease the score of s_k. Reminding $\boldsymbol{x_k}$ is always a positive instance, this behaviour is undesirable.

For $\mathrm{rank}^-(k)$, we define the following smooth surrogate H^- for H, shown in Fig 3b:

$$H^-(t) = \begin{cases} \sigma(\frac{t}{\tau}) & \text{if } t \leq 0, \quad \text{where } \sigma \text{ is the sigmoid function (Fig. 3c)} \\ \sigma(\frac{t}{\tau}) + 0.5 & \text{if } t \in [0;\delta] \quad \text{with } \delta \geq 0 \\ \rho \cdot (t - \delta) + \sigma(\frac{\delta}{\tau}) + 0.5 & \text{if } t > \delta \end{cases} \tag{4}$$

[1]For the sake of readability we drop in the following the dependence on $\boldsymbol{\theta}$ for the rank, *i.e.* $\mathrm{rank}(k) := \mathrm{rank}(k, \theta)$ and on the query for the similarity, *i.e.* $s_j := s(q_i, x_j)$.



(a) $H^+(x) = H(x)$ in Eq. (3) (b) $H^-(x)$ in Eq. (4) (c) Sigmoid used in [2]

Figure 3: Proposed surrogate losses for the Heaviside (step): with $H^+(x)$ in Fig. 3a and $H^-(x)$ in Fig. 3b, $\mathcal{L}_{\text{SupAP}}$ in Eq. (5) is an upper bound of $\mathcal{L}_{\text{AP}}$. In addition, $H^-(x)$ back-propagates gradients until the correct ranking is satisfied, in contrast to the sigmoid used in [2] (Fig. 3c).

where τ and ρ are hyperparameters, and δ is defined such that the sigmoidal part of H^- reaches the saturation regime and is fixed for the rest of the paper (see supplementary Sec. A). From the H^- smooth approximation defined in Eq. (4), we obtain the following smooth approximation $\text{rank}^-_s(k) = \sum_{j \in \mathcal{N}_i} H^-(s_j - s_k)$, leading to the following smooth AP loss approximation:

$$\mathcal{L}_{\text{SupAP}}(\boldsymbol{\theta}) = 1 - \frac{1}{M} \sum_{i=1}^{M} \frac{1}{|\mathcal{P}_i|} \sum_{k \in \mathcal{P}_i} \frac{\text{rank}^+(k)}{\text{rank}^+(k) + \text{rank}^-_s(k)} \tag{5}$$

$\mathcal{L}_{\text{SupAP}}$ in Eq. (5) fulfills two main features for AP optimization:

► (1) **$\mathcal{L}_{\text{SupAP}}$ is an upper bound of $\mathcal{L}_{\text{AP}}$ in Eq. (1).** Since H^- in Eq. (4) is an upper bound of a step function (Fig 3b), it is easy to see that $\mathcal{L}_{SupAP} \geq \mathcal{L}_{\text{AP}}$. This is a very important property, since it ensures that the model keeps training until the correct ranking is obtained. It is worth noting that existing smooth rank approximations in the literature [40, 3, 32, 2] do not fulfill this property.

► (2) **$\mathcal{L}_{\text{SupAP}}$ brings training gradients until the correct ranking plus a margin is fulfilled.** When the ranking is incorrect, the negative $\boldsymbol{x_j}$ is ranked before the positive $\boldsymbol{x_k}$, thus $s_j > s_k$ and $H^-(s_j - s_k)$ in Eq. (4) has a non-null derivative. We use a sigmoid to have a large gradient when $s_j - s_k$ is small. To overcome vanishing gradients of the sigmoid for large values $s_j - s_k$, we use a linear function ensuring constant ρ derivative. When the ranking is correct ($s_j < s_k$), we enforce robustness by imposing a margin parametrized by τ (sigmoid in Eq. (4)). This margin overcomes the brittleness of rank losses, which vanish as soon as the ranking is correct [15, 3, 28].

Comparison to SmoothAP [2] $\mathcal{L}_{\text{SupAP}}$ differs from $\mathcal{L}_{\text{SmoothAP}}$ in [2] by i) providing an upper bound on $\mathcal{L}_{\text{AP}}$, ii) improving the gradient flow (Fig. 3b vs Fig. 3c), and iii) overcoming adverse effects of the sigmoid for $rank^+$, as shown in Fig. 1a (and in supplementary sec. A). We experimentally verify the consistent gain brought out by $\mathcal{L}_{\text{SupAP}}$ over $\mathcal{L}_{\text{SmoothAP}}$.

3.2 Decomposable Average Precision

In Eq. (1), AP decomposes linearly between queries $\boldsymbol{q_i}$, but AP_i does not decomposes linearly between samples. We therefore focus our analysis of the non-decomposability on a single query. For a retrieval set Ω of N elements, we consider $\{\mathcal{B}^b\}_{b \in \{1:K\}}$ batches of size B, such that $N/B = K \in \mathbb{N}$. Let $\text{AP}^b_i(\boldsymbol{\theta})$ be the AP in batch b for query $\boldsymbol{q_i}$, we define the "decomposability gap" DG_{AP} as follows:

$$DG_{\text{AP}}(\boldsymbol{\theta}) = \frac{1}{K} \sum_{b=1}^{K} \text{AP}^b_i(\boldsymbol{\theta}) - \text{AP}_i(\boldsymbol{\theta}) \tag{6}$$

DG_{AP} in Eq. (6) is a direct measure of the non-decomposability of AP (see supplementary Sec. A). Our motivation here is to decrease DG_{AP}, *i.e.* to have the average AP over the batches as close as possible to the AP computed over the whole training set. To this aim, we introduce the following loss

during training:

$$\mathcal{L}_{\text{calibr.}}(\boldsymbol{\theta}) = \frac{1}{M}\sum_{i=1}^{M} \underbrace{\frac{1}{|\mathcal{P}_i|}\sum_{\boldsymbol{x_j}\in\mathcal{P}_i}[\alpha - s_j]_+}_{\mathcal{L}^+_{\text{calibr.}}} + \underbrace{\frac{1}{|\mathcal{N}_i|}\sum_{\boldsymbol{x_j}\in\mathcal{N}_i}[s_j - \beta]_+}_{\mathcal{L}^-_{\text{calibr.}}} \tag{7}$$

where $[x]_+ = \max(0, x)$. The loss $\mathcal{L}^+_{\text{calibr.}}$ enforces the score of the positive $\boldsymbol{x_i} \in \mathcal{P}_i$ to be larger than α, and $\mathcal{L}^-_{\text{calibr.}}$ enforces the score of the negative $\boldsymbol{x_j} \in \mathcal{N}_i$ to be smaller than $\beta < \alpha$. $\mathcal{L}_{\text{calibr.}}$ is a standard pair-based loss [12], which we revisit in our context to "calibrate" the values of the scores between mini-batches: intuitively, the fact that the positive (resp. negative) scores are above (resp. below) a threshold in the mini-batches makes the average AP closer to the AP on the whole dataset.

Upper bound on the decomposabilty gap To formalize this idea, we provide a theoretical analysis of the impact on the global ranking of $\mathcal{L}_{\text{calibr.}}$ in Eq. (7). Firstly, we can see that if $\mathcal{L}^-_{\text{calibr.}} = \mathcal{L}^+_{\text{calibr.}} = 0$, on each batch, the overall AP and the AP in batches is null, *i.e.* $DG_{\text{AP}}(\boldsymbol{\theta}) = 0$ and we get a decomposable AP. In a more general setting, we show that minimizing $\mathcal{L}_{\text{calibr.}}$ on each batch reduces the decomposability gap, hence improving the decomposability of the AP.

Let's consider K batches $\{\mathcal{B}^b\}_{b\in\{1:K\}}$ of batch size B divided in $\mathcal{P}_i^b$ positive instances and $\mathcal{N}_i^b$ negative instances w.r.t. the query $\boldsymbol{q_i}$. To give some insight we assume that the AP of each batch is one (*i.e.* $AP_i^b = 1$), and give the following upper bound of DG_{AP} :

$$0 \leq DG_{\text{AP}} \leq 1 - \frac{1}{\sum_{b=1}^{K}|\mathcal{P}_i^b|}\left(\sum_{b=1}^{K}\sum_{j=1}^{B}\frac{j + |\mathcal{P}_i^1| + \cdots + |\mathcal{P}_i^{b-1}|}{j + |\mathcal{P}_i^1| + \cdots + |\mathcal{P}_i^{b-1}| + |\mathcal{N}_i^1| + \cdots + |\mathcal{N}_i^{b-1}|}\right) \tag{8}$$

This upper bound of the decomposability gap is given in the worst case for the global AP : the global ranking is built from the juxtaposition of the batches (see supplementary Sec. A).

We can refine this upper bound by introducing the calibration loss $\mathcal{L}_{\text{calibr.}}$ and constraining the scores of positive and negative instances to be well calibrated. On each batch we define the following quantities $E_b^- = \sum_{j\in\mathcal{N}_i^-}\mathbb{1}(s_j > \beta)$ which are the negative instances that do not respect the constraints and $G_b^- = \sum_{j\in\mathcal{N}_i^-}\mathbb{1}(s_j \leq \beta)$ the negative instances that do. We similarly define E_b^+ and G_b^+. We then have the following upper bound on the decomposability gap :

$$0 \leq DG_{\text{AP}} \leq 1 - \frac{1}{\sum_{b=1}^{K}|\mathcal{P}_i^b|}\left(\sum_{b=1}^{K}\left[\sum_{j=1}^{G_b^+}\frac{j + G_1^+ + \cdots + G_{b-1}^+}{j + G_1^+ + \cdots + G_{b-1}^+ + E_1^- + \ldots E_{b-1}^-} + \right.\right. \tag{9}$$
$$\left.\left.\sum_{j=1}^{E_b^+}\frac{j + G_b^+ + |\mathcal{P}_i^1| + \cdots + |\mathcal{P}_i^{b-1}|}{j + G_b^+ + |\mathcal{P}_i^1| + \cdots + |\mathcal{P}_i^{b-1}| + |\mathcal{N}_i^1| + \cdots + |\mathcal{N}_i^{b-1}|}\right]\right)$$

This refined upper bound is tighter than the upper bound of Eq. (8). Our new $\mathcal{L}_{\text{calibr.}}$ loss directly optimizes this upper bound (by explicitly optimizing $E_b^-, E_b^+, E_b^+, G_b^+$), making it tighter, hence improving the decomposability of the AP (see supplementary Sec. A).

4 Experiments

Experimental setup We evaluate ROADMAP on the following three image retrieval datasets:
CUB-200-2011 [42] contains 11 788 images of birds classified into 200 fine-grained classes. We follow the standard protocol and use the first (resp. last) 100 classes for training (resp. evaluation).
Stanford Online Product (SOP) [36] is a dataset with 120 053 images of 22 634 objects classified into 12 categories (*e.g.* bikes, coffee makers). We use the reference train and test splits from [36].
INaturalist-2018 [41] is a large scale dataset of 461 939 wildlife animals images classified into 8142 classes. We use the splits from [2] with 70% of the classes in the train set and the rest in the test set.

ROADMAP settings For all experiments in Section 4.1 and Section 4.2, we use $\lambda = 0.5$ for $\mathcal{L}_{\text{ROADMAP}}$ in Eq. (2), $\tau = 0.01$ and $\rho = 100$ for $\mathcal{L}_{\text{SupAP}}$ in Eq. (5), $\alpha = 0.9$ and $\beta = 0.6$ for $\mathcal{L}_{\text{calibr.}}$

in Eq. (7). We study more in depth the impact of those parameters in Section 4.3. Deep models are trained using Adam [19] for ResNet-50 backbones and AdamW [21] for DeiT transformers [39].
Test protocol Methods are evaluated using the standard recall at k (R@k) and mean average precision at R [26] (mAP@R) metrics (see supplementary Sec. B).

4.1 ROADMAP validation

In this section, all models are trained in the same setting (ResNet-50 backbone, embedding size 512, batch size 64). The comparisons thus directly measures the impact of the training loss.

Comparison to AP approximations. In Table 1, we compare ROADMAP on the three datasets to recent AP loss approximations including the soft-binning approaches FastAP [3] and SoftBinAP [32], the generic solver BlackBox [33], and the smooth rank approximation [2]. We use the publicly available PyTorch implementations of all these baselines. We can see that ROADMAP outperforms all the current AP approximations by a large margin. The gain is especially pronounced on the large scale dataset INaturalist. This highlights the importance our two contributions, *i.e.* our robust smooth AP upper bound and our AP decomposability improvement (see supplementary Sec. B).

Table 1: Comparison between ROADMAP and state-of-the-art AP ranking based methods.

	CUB		SOP		INaturalist	
Method	R@1	mAP@R	R@1	mAP@R	R@1	mAP@R
FastAP [3]	58.9	22.9	78.2	51.3	53.5	19.6
SoftBin [32]	61.2	24.0	80.1	53.5	56.6	20.1
BlackBox [33]	62.6	23.9	80.0	53.1	52.3	15.2
SmoothAP [2]	62.1	23.9	80.9	54.6	59.8	20.7
ROADMAP	**64.2**	**25.3**	**82.0**	**56.5**	**64.5**	**25.1**

Comparison to memory methods.

XBM stores the embeddings of previously seen batches to alleviate complex batch sampling and better approximate AP on the whole dataset. Although XBM has a low memory overhead (a few hundreds megabytes on SOP), it is time consuming. We ran experiments storing the entire dataset for SOP (60k embeddings), but for INaturalist we could not train while storing all the dataset in tractable time. We chose to store the same amount of embeddings as for SOP : 60k embeddings which is about 17% of the training set.

We can see in Table 2 that XBM is approximately 3 times longer to train than ROADMAP. This becomes critical on INaturalist, where training while storing 60k images takes about 3 days, and reaches only a R@1 of 60. Consequently, ROADMAP outperforms XBM on both datasets; there is a $\sim$+2pt increase on both metrics for SOP and an especially large gap on INaturalist. In the latter, not being able to store all the embeddings affects drastically the performances of the XBM in a negative way. There is a 5pt difference in R@1 and more than 6pt in mAP@R. This demonstrates the suitability of ROADMAP on large-scale settings.

Table 2: Our method compared to cross batch memory [44]. The unit of time is m/e which stands for minutes per epoch.

	SOP			INaturalist		
Method	R@1	mAP@R	time$\downarrow$	R@1	mAP@R	time$\downarrow$
XBM [44]	80.6	54.9	6	59.3	18.5	34
ROADMAP (ours)	**82.0**	**56.5**	**2**	**64.5**	**25.1**	**12**

Ablation study. To study more in depth the impact of our contributions, we perform ablation studies in Table 3. We show the improvement against SmoothAP [2] when changing the sigmoid by H^+ and H^- for $\mathcal{L}_{\text{SupAP}}$ in Eq. (5), and the use of $\mathcal{L}_{\text{calibr.}}$ in Eq. (7). We can see that $\mathcal{L}_{\text{SupAP}}$ consistently improves performances over $\mathcal{L}_{\text{SmoothAP}}$ (0.9pt on CUB, 0.5pt on SOP and 1.5pt on INaturalist). $\mathcal{L}_{\text{SupAP}}$ and $\mathcal{L}_{\text{calibr.}}$ equally contribute to the overall gain in CUB and SOP, but the gain of $\mathcal{L}_{\text{calibr.}}$ is much

more important on INaturalist. This is explained by the fact that the batch vs. dataset ratio size $\frac{B}{N}$ is tiny ($\ll 1$), making the decomposability gap in Eq. (6) huge. We can see that $\mathcal{L}_{\text{calibr.}}$ is very effective for reducing this gap and brings a gain of more than 3pt.

Table 3: Ablation study for the impact of our two contribution on and the SmoothAP baseline.

			CUB		SOP		INaturalist	
Method	H^-	$\mathcal{L}_{\text{calibr.}}$	R@1	mAP@R	R@1	mAP@R	R@1	mAP@R
SmoothAP [2]	✗	✗	62.1	23.9	80.9	54.6	59.7	20.7
SupAP	✓	✗	62.9	24.6	81.4	55.3	61.2	21.3
ROADMAP	✓	✓	**64.2**	**25.3**	**82.0**	**56.5**	**64.5**	**25.1**

4.2 State of the art comparison

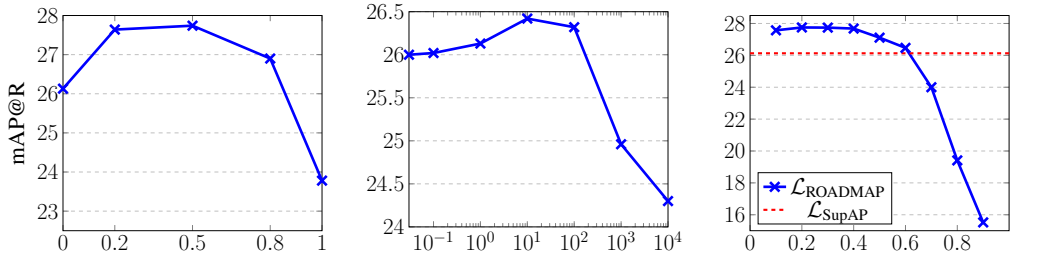
We compare ROADMAP to other state of the art methods across three image retrieval datasets and report the results in Table 4. We divide competitor methods into three categories: metric learning [34, 43, 52, 17, 44, 48], classification losses for image retrieval [53, 51, 1, 38], and AP approximations [3, 33, 2]. ROADMAP falls in the latter category. We use the same setup as in Section 4.1 and follow standard practices for ResNet-50 [38, 48, 1] by using larger images (256×256 on SOP and CUB) and using max instead of average pooling and layer normalization for CUB.

Using the popular ResNet-50 backbone, ROADMAP establishes a new state of the art across all methods for SOP and the challenging INaturalist dataset and outperforms all previous AP approximations on CUB, while being competitive with the other two top performers (ProxyNCA++ and SEC). R@k improvements are consistent on all datasets with a $\sim$2pts R@1 increase on INaturalist and $\sim$3pts increase on SOP compared to SmoothAP, the best performing AP approximation from the literature.

Switching the backbone to the more recent vision transformer architecture DeiT [5, 39], further lifts the performances of ROADMAP by several point, from 3 to 9 points depending on the dataset, with a smaller embedding size (384 *vs* 512). The decomposable AP approximation ROADMAP also outperforms by a significant margin IRT_R, the DeiT architecture for image retrieval introduced in [7] trained with a contrastive loss. Overall ROADMAP achieves state-of-the-art performances across all three datasets by a significant margin.

4.3 Model Analysis

We show in Fig. 4 the impact of the main ROADMAP hyperparameters on INaturalist. The relative weighting λ from Eq. (2) controls the balance between our two training objectives $\mathcal{L}_{\text{SupAP}}$ and $\mathcal{L}_{\text{calibr.}}$: $\lambda = 0$ reduces $\mathcal{L}_{\text{ROADMAP}}$ to $\mathcal{L}_{\text{SupAP}}$ while $\lambda = 1$ to $\mathcal{L}_{\text{calibr.}}$. We can see in Fig. 4a that training with the complete $\mathcal{L}_{\text{ROADMAP}}$ with both $\mathcal{L}_{\text{calibr.}}$ and $\mathcal{L}_{\text{SupAP}}$ is always better than using only one of the two losses. Note that results are stable in the $[0.2, 0.8]$ range with a consistent $\sim$1.5pt increase, demonstrating the robustness of ROADMAP to this hyperparameter tuning.



(a) mAP@R *vs* λ for $\mathcal{L}_{\text{ROADMAP}}$ (b) mAP@R *vs* ρ for $\mathcal{L}_{\text{SupAP}}$ (c) mAP@R *vs* $\alpha - \beta$ for $\mathcal{L}_{\text{calibr.}}$

Figure 4: Analysis of ROADMAP hyperparameters on INaturalist (batch size 224).

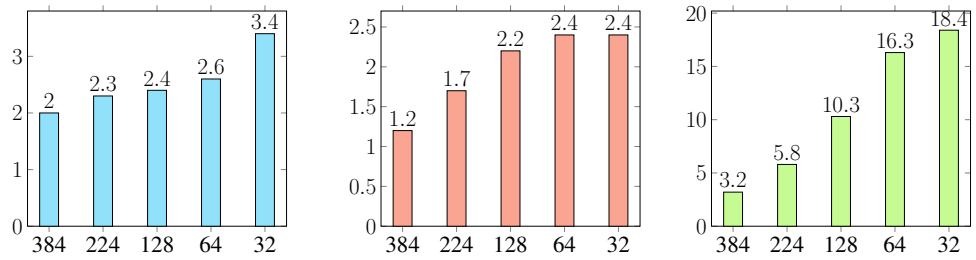
Fig. 4b shows the influence of the slope ρ that controls the linear regime in H^- and determines the amount of gradient backpropagated for negative samples with a (wrong) high score. As shown in

Table 4: Comparison of state of the art performances from the literature on SOP, CUB and INaturalist with the proposed ROADMAP (recall@k). Except for the DeiT category, all methods rely on a standard convolutional backbone (generally ResNet-50).

			SOP			CUB				INaturalist			
	Method	dim	1	10	100	1	2	4	8	1	4	16	32
Metric learning	Triplet SH [46]	512	72.7	86.2	93.8	63.6	74.4	83.1	90.0	58.1	75.5	86.8	90.7
	LiftedStruct [36]	512	62.1	79.8	91.3	47.2	58.9	70.2	80.2	-	-	-	-
	MIC [34]	512	77.2	89.4	95.6	66.1	76.8	85.6	-	-	-	-	-
	MS [43]	512	78.2	90.5	96.0	65.7	77.0	86.3	91.2	-	-	-	-
	SEC [52]	512	78.7	90.8	96.6	68.8	79.4	87.2	92.5	-	-	-	-
	HORDE [17]	512	80.1	91.3	96.2	66.8	77.4	85.1	91.0	-	-	-	-
	XBM [44]	128	80.6	91.6	96.2	65.8	75.9	84.0	89.9	-	-	-	-
	Triplet SCT [48]	512/64	81.9	92.6	96.8	57.7	69.8	79.6	87.0	-	-	-	-
Classification	ProxyNCA [25]	512	73.7	-	-	49.2	61.9	67.9	72.4	61.6	77.4	87.0	90.6
	ProxyGML [53]	512	78.0	90.6	96.2	66.6	77.6	86.4	-	-	-	-	-
	NSoftmax [51]	512	78.2	90.6	96.2	61.3	73.9	83.5	90.0	-	-	-	-
	NSoftmax [51]	2048	79.5	91.5	96.7	65.3	76.7	85.4	91.8	-	-	-	-
	Cross-Entropy [1]	2048	81.1	91.7	96.3	69.2	79.2	86.9	91.6	-	-	-	-
	ProxyNCA++ [38]	512	80.7	92.0	96.7	69.0	79.8	87.3	92.7	-	-	-	-
	ProxyNCA++ [38]	2048	81.4	92.4	96.9	72.2	82.0	89.2	93.5	-	-	-	-
AP loss	FastAP [3]	512	76.4	89.0	95.1	-	-	-	-	60.6	77.0	87.2	90.6
	BlackBox [33]	512	78.6	90.5	96.0	64.0	75.3	84.1	90.6	62.9	79.4	88.7	91.7
	SmoothAP [2]	512	80.1	91.5	96.6	-	-	-	-	67.2	81.8	90.3	93.1
	SoftBin* [32]	512	80.6	91.3	96.1	61.2	73.14	83.0	89.5	64.2	77.1	82.7	91.7
	ROADMAP (ours)	512	83.1	92.7	96.3	68.5	78.7	86.6	91.9	69.1	83.1	91.3	93.9
DeiT	IRT_R [7]	384	84.2	93.7	97.3	76.6	85.0	91.1	94.3	-	-	-	-
	ROADMAP (ours)	384	**86.0**	**94.4**	**97.6**	**77.4**	**85.5**	**91.4**	**95.0**	**73.6**	**86.2**	**93.1**	**95.2**

Fig. 4b, the improvement is important and stable in $[10, 100]$. Note that $\rho > 0$ already improves the results compared to $\rho = 0$ in [2]. There is an important decrease when $\rho \gg 100$ probably due to the high gradient that takes over the signal for correctly ranked samples.
The impact of the margin $\alpha - \beta$ in $\mathcal{L}_{\text{calibr.}}$ is shown in Fig. 4c. Once again, ROADMAP exhibits a robust behaviour w.r.t. the values of its hyperparameters: any margin in the $[0.1, 0.6]$ range results in an improvement in mAP@R compared to the $\mathcal{L}_{\text{SupAP}}$ baseline without the decomposability loss. Best results are achieved with smaller margins $0.1 < \alpha - \beta < 0.4$.

Fig. 5 shows the improvement in mAP@R on the three datasets when adding $\mathcal{L}_{\text{calibr.}}$ to $\mathcal{L}_{\text{SupAP}}$. We can see that the increase becomes larger as the batch size gets smaller. This confirms our intuition that the decomposability in $\mathcal{L}_{\text{calibr.}}$ has a stronger effect on smaller batch sizes, for which the AP estimation is noisier and DG_{AP} larger. This is critical on the large-scale dataset INaturalist where the batch AP on usual batch sizes is a very poor approximation of the global AP.



(a) CUB (b) SOP (c) INaturalist

Figure 5: Relative increase of the mAP@R *vs* batch size when adding $\mathcal{L}_{\text{calibr.}}$ to $\mathcal{L}_{\text{SupAP}}$.

As a qualitative assessment, we show in Fig. 6 some results of ROADMAP on INaturalist. We show the queries (in purple) and the 4 most similar retrieved images (in green). We can appreciate the semantic quality of the retrieval. More qualitative results are provided in supplementary Sec. C.

Fig. 7 shows another qualitative assessment on INaturalist, where ROADMAP corrects some failing cases of the SmoothAP baseline.

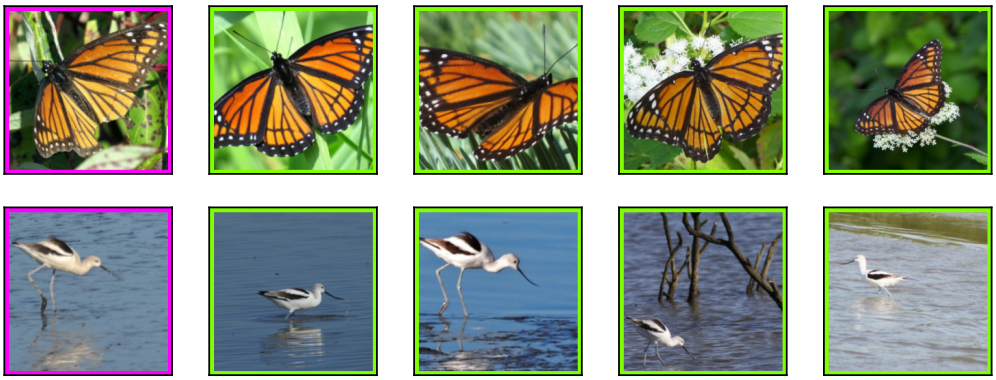


Figure 6: Results on INaturalist: a query (purple) with the 4 most similar retrieved images (green).

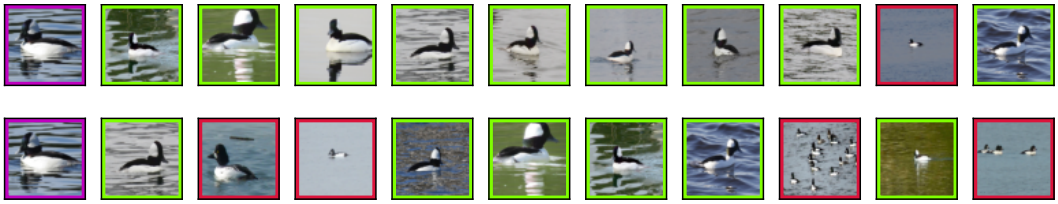


Figure 7: Results on INaturalist: a query (purple) with the 9 most similar retrieved images, green for relevant images, red otherwise. Top line results with ROADMAP. Bottom line results with SmoothAP.

5 Conclusion

This paper introduces the ROADMAP method for gradient-based optimization of average precision. ROADMAP is based on a smooth rank approximation, leading to the $\mathcal{L}_{\text{SupAP}}$ being both accurate and robust. To overcome the lack of decomposability in AP, ROADMAP is equipped with a calibration loss $\mathcal{L}_{\text{calibr.}}$ which aims at reducing the decomposability gap. We provide theoretical guarantees as well as experiments to assess this behavior. Experiments show that ROADMAP can combine the strength of ranking methods with the simplicity of a batch strategy. Without bells and whistles, ROADMAP is able to outperform state-of-the-art performances on three datasets, and remains effective even with small batch sizes.

As any work on image retrieval, our contribution could be applied to critical applications in surveillance scenarios, *e.g.* face recognition or person re-identification. ROADMAP is neither worse nor better than previous work in this regard. Our work is also a data-driven learning method, and thus inherits the risk of perpetuating dataset biases. Future work will focus on improving fair and accurate retrieval by reducing dataset biases. We also plan to relax the need for full supervision to tackle situations more representative to in-the-wild scenarios.

Acknowledgement This work was done under a grant from the the AHEAD ANR program (ANR-20-THIA-0002). It was granted access to the HPC resources of IDRIS under the allocation 2021-AD011012645 made by GENCI.

References

[1] Malik Boudiaf, Jérôme Rony, Imtiaz Masud Ziko, Eric Granger, Marco Pedersoli, Pablo Piantanida, and Ismail Ben Ayed. A unifying mutual information view of metric learning: cross-entropy vs. pairwise losses. In *European Conference on Computer Vision*, pages 548–564. Springer, 2020.

[2] Andrew Brown, Weidi Xie, Vicky Kalogeiton, and Andrew Zisserman. Smooth-ap: Smoothing the path towards large-scale image retrieval. In *European Conference on Computer Vision*, pages 677–694. Springer, 2020.

[3] Fatih Cakir, Kun He, Xide Xia, Brian Kulis, and Stan Sclaroff. Deep metric learning to rank. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1861–1870, 2019.

[4] Micael Carvalho, Rémi Cadène, David Picard, Laure Soulier, Nicolas Thome, and Matthieu Cord. Cross-modal retrieval in the cooking context: Learning semantic text-image embeddings. In Kevyn Collins-Thompson, Qiaozhu Mei, Brian D. Davison, Yiqun Liu, and Emine Yilmaz, editors, *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, pages 35–44. ACM, 2018.

[5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[6] Thibaut Durand, Nicolas Thome, and Matthieu Cord. Exploiting negative evidence for deep latent structured models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(2):337–351, 2019.

[7] Alaaeldin El-Nouby, Natalia Neverova, Ivan Laptev, and Hervé Jégou. Training vision transformers for image retrieval. *arXiv preprint arXiv:2102.05644*, 2021.

[8] Martin Engilberge, Louis Chevallier, Patrick Perez, and Matthieu Cord. Sodeep: A sorting deep net to learn ranking loss surrogates. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[9] Fartash Faghri, David J. Fleet, Jamie Ryan Kiros, and Sanja Fidler. VSE++: improving visual-semantic embeddings with hard negatives. In *British Machine Vision Conference 2018, BMVC 2018, Newcastle, UK, September 3-6, 2018*, page 12. BMVA Press, 2018.

[10] Weifeng Ge. Deep metric learning with hierarchical triplet loss. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[11] Albert Gordo, Jon Almazán, Jérôme Revaud, and Diane Larlus. End-to-end learning of deep visual representations for image retrieval. *Int. J. Comput. Vis.*, 124(2):237–254, 2017.

[12] 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.

[13] Ben Harwood, Vijay Kumar B G, Gustavo Carneiro, Ian Reid, and Tom Drummond. Smart mining for deep metric learning. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. corr abs/1512.03385 (2015), 2015.

[15] Kun He, Fatih Cakir, Sarah Adel Bargal, and Stan Sclaroff. Hashing as tie-aware learning to rank. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[16] Kun He, Yan Lu, and Stan Sclaroff. Local descriptors optimized for average precision. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[17] Pierre Jacob, David Picard, Aymeric Histace, and Edouard Klein. Metric learning with horde: High-order regularizer for deep embeddings. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6539–6548, 2019.

[18] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*, 2017.

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

[20] Marc T. Law, Nicolas Thome, and Matthieu Cord. Learning a distance metric from relative comparisons between quadruplets of images. *Int. J. Comput. Vis.*, 121(1):65–94, 2017.

[21] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[22] R. Manmatha, Chao-Yuan Wu, Alexander J. Smola, and Philipp Krähenbühl. Sampling matters in deep embedding learning. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2859–2867. IEEE Computer Society, 2017.

[23] Brian Mcfee and Gert Lanckriet. Metric learning to rank. In *In Proceedings of the 27th annual International Conference on Machine Learning (ICML*, 2010.

[24] Pritish Mohapatra, Michal Rolínek, C.V. Jawahar, Vladimir Kolmogorov, and M. Pawan Kumar. Efficient optimization for rank-based loss functions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[25] Yair Movshovitz-Attias, Alexander Toshev, Thomas K Leung, Sergey Ioffe, and Saurabh Singh. No fuss distance metric learning using proxies. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 360–368, 2017.

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

[27] Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. Pytorch metric learning, 2020.

[28] Marin Vlastelica P., Anselm Paulus, Vít Musil, Georg Martius, and Michal Rolínek. Differentiation of blackbox combinatorial solvers. In *ICLR*, 2020.

[29] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

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

[31] Filip Radenovic, Giorgos Tolias, and Ondrej Chum. CNN image retrieval learns from bow: Unsupervised fine-tuning with hard examples. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part I*, volume 9905 of *Lecture Notes in Computer Science*, pages 3–20. Springer, 2016.

[32] Jerome Revaud, Jon Almazán, Rafael S Rezende, and Cesar Roberto de Souza. Learning with average precision: Training image retrieval with a listwise loss. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5107–5116, 2019.

[33] Michal Rolínek, Vít Musil, Anselm Paulus, Marin Vlastelica, Claudio Michaelis, and Georg Martius. Optimizing rank-based metrics with blackbox differentiation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7620–7630, 2020.

[34] Karsten Roth, Biagio Brattoli, and Bjorn Ommer. Mic: Mining interclass characteristics for improved metric learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8000–8009, 2019.

[35] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

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

[37] Yumin Suh, Bohyung Han, Wonsik Kim, and Kyoung Mu Lee. Stochastic class-based hard example mining for deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[38] 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.

[39] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020.

[40] Evgeniya Ustinova and Victor Lempitsky. Learning deep embeddings with histogram loss. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[41] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8769–8778, 2018.

[42] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[43] Xun Wang, Xintong Han, Weilin Huang, Dengke Dong, and Matthew R Scott. Multi-similarity loss with general pair weighting for deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5022–5030, 2019.

[44] 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.

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

[46] 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.

[47] Eric Xing, Michael Jordan, Stuart J Russell, and Andrew Ng. Distance metric learning with application to clustering with side-information. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15. MIT Press, 2003.

[48] Hong Xuan, Abby Stylianou, Xiaotong Liu, and Robert Pless. Hard negative examples are hard, but useful. In *European Conference on Computer Vision*, pages 126–142. Springer, 2020.

[49] Omry Yadan. Hydra - a framework for elegantly configuring complex applications. Github, 2019.

[50] Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims. A support vector method for optimizing average precision. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 271–278, New York, NY, USA, 2007. ACM.

[51] Andrew Zhai and Hao-Yu Wu. Making classification competitive for deep metric learning. *CoRR*, abs/1811.12649, 2018.

[52] Dingyi Zhang, Yingming Li, and Zhongfei Zhang. Deep metric learning with spherical embedding. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 18772–18783. Curran Associates, Inc., 2020.

[53] Yuehua Zhu, Muli Yang, Cheng Deng, and Wei Liu. Fewer is more: A deep graph metric learning perspective using fewer proxies. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 17792–17803. Curran Associates, Inc., 2020.

A ROADMAP model

A.1 Properties of SupAP & comparison to SmoothAP

We further discuss and give additional explanations of the property of our $\mathcal{L}_{\text{SupAP}}$ loss function, and especially its comparison with respect to the SmoothAP [2] baseline.

As shown in Fig. 1.a of the main paper, and discussed in Section 3.1 ("Comparison to SmoothAP"), the smooth rank approximation in [2] has several drawbacks, that we show below:

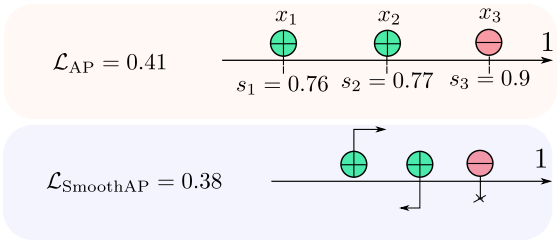


Figure 8: Limitation of the smooth rank approximation in [2]: contradictory gradient flow for the positives samples $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ (in green), vanishing gradient for the negative example $\boldsymbol{x_3}$ (in red), and no guarantees of having an upper bound of $\mathcal{L}_{\text{AP}}$.

Specifically, we explain in more detail the following three limitations identified in the main paper for SmoothAP [2], which comes from the use of the sigmoid function to approximate the Heaviside (step) function for computing the rank:

i **Contradictory gradient flow for positives samples:** Firstly we can see on the toy dataset of Fig. 8 that the gradients of the two positive examples (in green) with SmoothAP have opposite directions. The positive with the lowest rank $\boldsymbol{x_1}$ has a gradient in the good direction, since it leads to increase $\boldsymbol{x_1}$'s score because the correct ordering is not reached (the negative instance $\boldsymbol{x_3}$ has a better rank). But the gradient of the positive with the highest rank $\boldsymbol{x_2}$ is on the wrong direction, since it tends to decrease $\boldsymbol{x_2}$'s score. This is an undesirable behaviour, which comes from the use of the sigmoid in $\mathcal{L}_{\text{SmoothAP}}$. In the example of Fig. 8, we can actually show that

$$\boxed{\frac{\partial \mathcal{L}_{\text{SmoothAP}}}{\partial s_1} = -\frac{\partial \mathcal{L}_{\text{SmoothAP}}}{\partial s_2}}$$

To see this we write :

$$\begin{aligned}\frac{\partial \mathcal{L}_{\text{SmoothAP}}}{\partial s_1} =& \frac{\partial \mathcal{L}_{\text{SmoothAP}}}{\partial \operatorname{rank}^+(x_1)} \cdot \frac{\partial \operatorname{rank}^+(x_1)}{\partial s_1} + \frac{\partial \mathcal{L}_{\text{SmoothAP}}}{\partial \operatorname{rank}^+(x_2)} \cdot \frac{\partial \operatorname{rank}^+(x_2)}{\partial s_1} \\ &+ \frac{\partial \mathcal{L}_{\text{SmoothAP}}}{\partial \operatorname{rank}^-(x_1)} \cdot \frac{\partial \operatorname{rank}^-(x_1)}{\partial s_1} + \frac{\partial \mathcal{L}_{\text{SmoothAP}}}{\partial \operatorname{rank}^-(x_2)} \cdot \frac{\partial \operatorname{rank}^-(x_2)}{\partial s_1}\end{aligned}$$

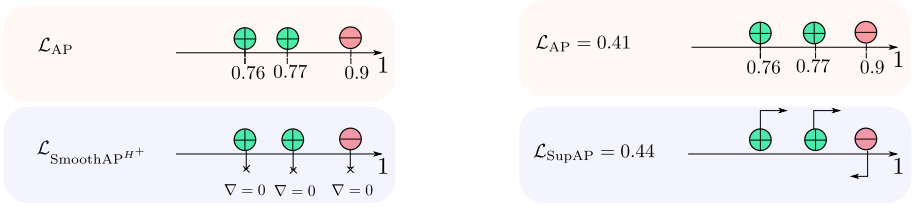
Because $\operatorname{rank}^-(x_2) = \sigma(\frac{s_3 - s_2}{\tau})$, we have $\frac{\partial \operatorname{rank}^-(x_2)}{\partial s_1} = 0$ and $\frac{\partial \operatorname{rank}^-(x_1)}{\partial s_1} = 0$ in the example of Fig. 8, because $\operatorname{rank}^-(x_1) = \sigma(\frac{s_3 - s_1}{\tau})$ and $s_3 - s_1$ falls into the saturation regime of the sigmoid. We get a similar result for the derivative of $\mathcal{L}_{\text{SmoothAP}}$ wrt. s_2 :

$$\frac{\partial \mathcal{L}_{\text{SmoothAP}}}{\partial s_2} = \frac{\partial \mathcal{L}_{\text{SmoothAP}}}{\partial \operatorname{rank}^+(x_1)} \cdot \frac{\partial \operatorname{rank}^+(x_1)}{\partial s_2} + \frac{\partial \mathcal{L}_{\text{SmoothAP}}}{\partial \operatorname{rank}^+(x_2)} \cdot \frac{\partial \operatorname{rank}^+(x_2)}{\partial s_2}$$

Furthermore we have :

$$\frac{\partial \operatorname{rank}^+(x_1)}{\partial s_1} = -\frac{\partial \operatorname{rank}^+(x_1)}{\partial s_2}$$

Indeed $\operatorname{rank}^+(x_1) = 1 + \sigma(\frac{s_2 - s_1}{\tau})$, such that $\frac{\partial \operatorname{rank}^+(x_1)}{\partial s_1} = -\tau \cdot \sigma(\frac{s_2 - s_1}{\tau}) \left(1 - \sigma(\frac{s_2 - s_1}{\tau})\right)$ and $\frac{\partial \operatorname{rank}^+(x_1)}{\partial s_2} = \tau \cdot \sigma(\frac{s_2 - s_1}{\tau}) \left(1 - \sigma(\frac{s_2 - s_1}{\tau})\right)$. Similarly the derivatives of $\operatorname{rank}^+(x_2)$ wrt. s_1



(a) When replacing H^+ by the Heaviside function in SmoothAP we stop the unexpected behaviour of the gradient flow. However there is still vanishing gradients.

(b) Our $\mathcal{L}_{\text{SupAP}}$ has gradients that do not stop until the correct ranking is achieved.

Figure 9: We illustrates the different steps to built $\mathcal{L}_{\text{SupAP}}$. On Fig. 9a we change H^+ to be the true Heaviside (step) function. On Fig. 9b we replace the sigmoid by H^- defined in Eq. (4) of the main paper. Using H^+ and H^-, $\mathcal{L}_{\text{SupAP}}$ is an upper bound of $\mathcal{L}_{\text{AP}}$.

and s_2 also have opposite signs: $\frac{\partial \, \text{rank}^+(x_2)}{\partial s_1} = -\frac{\partial \, \text{rank}^+(x_2)}{\partial s_2}$. It concludes the proof that $\frac{\partial \mathcal{L}_{\text{SmoothAP}}}{\partial s_1} = -\frac{\partial \mathcal{L}_{\text{SmoothAP}}}{\partial s_2}$.

ii **Vanishing gradients:** Secondly, SmoothAP [2] has vanishing gradients due to its use of the sigmoid function. This is illustrated on the toy dataset in Fig. 8. The negative instance $\boldsymbol{x_3}$ has a high score s_3, but does not receive any gradient, which does not enable it to lower its score although it would improve the overall ranking. This is because the score difference between $\boldsymbol{x_3}$ and $\boldsymbol{x_2}$ is large, *i.e.* $s_3 - s_2 = 0.13$. Similarly, $s_3 - s_1 = 0.14$. Consequently, both $s_3 - s_2$ and $s_3 - s_1$ fall into the saturation regime of the sigmoid, preventing to propagate any gradient (see Fig. 3c. in the main paper).

iii **Finally, $\mathcal{L}_{\text{SmoothAP}}$ is not an upper bound of $\mathcal{L}_{\text{AP}}$.** The use of the sigmoid means that both rank^+ and rank^- can be over or under estimated. If rank^+ is overestimated (resp. underestimated) $\mathcal{L}_{\text{SmoothAP}}$ underestimates $\mathcal{L}_{\text{AP}}$ (resp. overestimates). And if rank^- is overestimated (resp. underestimated) $\mathcal{L}_{\text{SmoothAP}}$ overestimates $\mathcal{L}_{\text{AP}}$ (resp. overestimated). Therefore, $\mathcal{L}_{\text{SmoothAP}}$ can be larger or lower than $\mathcal{L}_{\text{AP}}$ in general. In the example of Fig. 8, we show that $\mathcal{L}_{\text{SmoothAP}}$ is lower than $\mathcal{L}_{\text{AP}}$.

We address those three issues with $\mathcal{L}_{\text{SupAP}}$:

i **Using the the true Heaviside (step) function $\mathbf{H}^+$ for $\mathbf{rank}^+$** allows to have the expected behaviour regarding the gradients of positives. When Changing $\mathbf{H}^+$ for rank^+ in Fig. 9a, we can see that we fix the problem of opposite gradients for the positive examples $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ - although the gradient is zero.

ii **Using $\mathbf{H}^-$ for $\mathbf{rank}^-$ overcomes vanishing gradients**. By using $\mathbf{H}^-$ in Eq. (4) in submission, we design a linear function for positive $(s_j - s_k)$ values, where s_j (resp. s_k) is the score of a negative (resp. positive) example - see Fig. 3b in the main paper. We can see in Fig. 9b that this change enables to have gradients in the correct directions for the two positive instances $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ (tending to increase their scores), and for the negative instance $\boldsymbol{x_3}$ (tending to decrease its score).

iii **$\mathcal{L}_{\text{SupAP}}$ is an upper bound of $\mathcal{L}_{\text{AP}}$.** By the proposed design of $\mathbf{H}^-$ in Eq. (4) in submission, we have $\text{rank}^-_s(k) \geq \text{rank}^-(k)$. Since we do not approximate $\text{rank}^+(k)$ by keeping the Heaviside function, it leads to $\frac{\text{rank}^+(k)}{\text{rank}^+(k)+\text{rank}^-_s(k)} \leq \frac{\text{rank}^+(k)}{\text{rank}^+(k)+\text{rank}^-(k)}$, and therefore $\boldsymbol{\mathcal{L}_{\text{SupAP}} \geq \mathcal{L}_{\text{AP}}}$.

Overall, $\mathcal{L}_{\text{SupAP}}$ has all the desired properties : i) A correct gradient flow during training, ii) No vanishing gradients while the correct ranking is not reached, iii) Being an upper bound on the AP loss $\mathcal{L}_{\text{AP}}$.

A.2 Properties of the $\mathcal{L}_{\text{calibr.}}$ loss function

We remind the reader of the definition of the decomposability gap given in Eq. (6) of the main paper.

$$DG_{\text{AP}}(\boldsymbol{\theta}) = \frac{1}{K}\sum_{b=1}^{K} \text{AP}_i^b(\boldsymbol{\theta}) - \text{AP}_i(\boldsymbol{\theta})$$

We illustrates the decomposability gap, DG_{AP} with the toy dataset of Fig. 10. The decomposability gap comes from the fact that the AP is not decomposable in mini-batches as we discuss in the Sec. 3.2 of the main paper. The motivation behind $\mathcal{L}_{\text{calibr.}}$ is thus to force the scores of the different batches to aligned as illustrated in the Fig. 2b of the main paper.

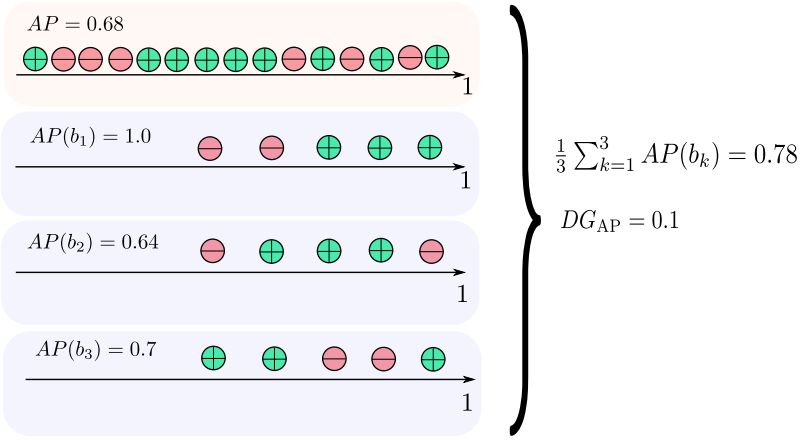


Figure 10: Illustration of the decomposability gap on a toy dataset.

Proof of Eq. (8): Upper bound on the DG_{AP} with no $\mathcal{L}_{\text{AP}}$ We choose a setting for the proof of the upper bound similar to the one used for training, *i.e.* all the batch have the same size, and the number of positive instances per batch (*i.e.* $\mathcal{P}_i^b$) is the same.

Eq. (8) from the main paper gives an upper bound for DG_{AP}. This upper bound is given in the worst case: when the AP has the lowest value guaranteed by the AP on each batch. We illustrate this case in Fig. 11.

In Eq. (8) from the main paper the 1 in the right hand term comes from the average of AP over all batches:

$$\frac{1}{K}\sum_{b=1}^{K} AP_i^b(\theta) = 1$$

We then justify the term in the parenthesis of Eq. (8) in the main paper, which is the lower bound of the AP. In the global ordering the positive instances are ranked after all the positive instances from previous batches giving the following rank^+: $j + |\mathcal{P}_i^1| + \cdots + |\mathcal{P}_i^{b-1}|$, with j the rank^+ in the batch, Positive instances are also ranked after all negative instances from previous batches giving rank^-: $|\mathcal{N}_i^1| + \cdots + |\mathcal{N}_i^{b-1}|$.

Therefore we obtain the resulting upper bound of Eq. (8) of the main paper:

$$0 \leq DG_{\text{AP}} \leq 1 - \frac{1}{\sum_{b=1}^{K} |\mathcal{P}_i^b|}\left(\sum_{b=1}^{K}\sum_{j=1}^{B} \frac{j + |\mathcal{P}_i^1| + \cdots + |\mathcal{P}_i^{b-1}|}{j + |\mathcal{P}_i^1| + \cdots + |\mathcal{P}_i^{b-1}| + |\mathcal{N}_i^1| + \cdots + |\mathcal{N}_i^{b-1}|}\right)$$

Proof of Eq. (9): Upper bound on the DG_{AP} with $\mathcal{L}_{\text{AP}}$ In the main paper we refine the upper bound on DG_{AP} in Eq. (9) by adding $\mathcal{L}_{\text{calibr.}}$ which calibrates the absolute scores across the mini-batches.

We now write that each positive instance that respects the constraint of $\mathcal{L}_{\text{calibr.}}$ is ranked after the positive instances of previous batch that respect the constraint giving the following rank^+:

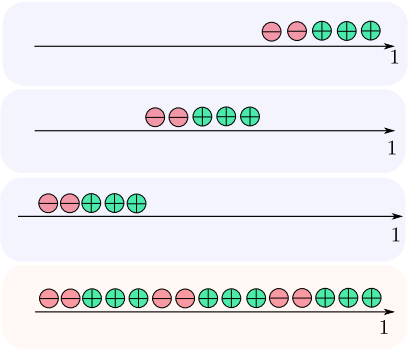


Figure 11: The worst case when computing the global AP would be that each batch is juxtaposed.

$j + G_1^+ + \cdots + G_{b-1}^+$, with j the rank^+ in the current batch. Positive instances are also ranked after the negative instances of previous batches that do not respect the constraints yielding rank^- : $E_1^- + \cdots + E_{b-1}^-$.

We then write that positive instances that do not respect the constraints are ranked after all positive instances from previous batches and the positive instances respecting the constraints of the current batch giving $\mathrm{rank}^+ : \ j + G_b^+ |\mathcal{P}_i^1| + \cdots + |\mathcal{P}_i^{b-1}|$. They also are ranked after all the negative instances from previous batches giving $\mathrm{rank}^- : |\mathcal{N}_i^1| + \cdots + |\mathcal{N}_i^{b-1}|$.

Resulting in Eq. (9) from the main paper:

$$0 \leq DG_{\text{AP}} \leq 1 - \frac{1}{\sum_{b=1}^{K} |\mathcal{P}_i^b|} \left(\sum_{b=1}^{K} \left[\sum_{j=1}^{G_b^+} \frac{j + G_1^+ + \cdots + G_{b-1}^+}{j + G_1^+ + \cdots + G_{b-1}^+ + E_1^- + \ldots E_{b-1}^-} + \right. \right.$$
$$\left. \left. \sum_{j=1}^{E_b^+} \frac{j + G_b^+ + |\mathcal{P}_i^1| + \cdots + |\mathcal{P}_i^{b-1}|}{j + G_b^+ + |\mathcal{P}_i^1| + \cdots + |\mathcal{P}_i^{b-1}| + |\mathcal{N}_i^1| + \cdots + |\mathcal{N}_i^{b-1}|} \right] \right)$$

A.3 Choice of δ

In the main paper we introduce δ in Eq. (4) to define H^-. We choose δ as the point where the gradient of the sigmoid function becomes low $< \epsilon$, and we then have $\delta = \tau \cdot \ln \frac{1-\epsilon}{\epsilon}$. This is illustrated in Fig. 12. For our experiments we use $\epsilon = 10^{-2}$ giving $\delta \simeq 0.05$.

B Experiments

B.1 Metrics

We detail here the performance metrics that we use to evaluate our models.

Recall@K The Recall@K metrics (Eq. (10)) is often used in the literature. For a single query the Recall@K is 1 if a positive instance is in the K nearest neighbors, and 0 otherwise. The Recall@K is then averaged on all the queries. Researcher use different values of K for a given dataset (*e.g.* 1, 2, 4, 8 on CUB).

$$R@K = \frac{1}{M} \sum_{i=1}^{M} r(i), \quad \text{where } r(i) = \begin{cases} 1 \text{ if a positive instance has a ranking smaller than i} \\ 0 \quad \text{otherwise} \end{cases} \tag{10}$$

mAP@R Recently, the mAP@R (Eq. (11)) has been introduced in [26]. The authors show that this metric is less noisy and better captures the performance of a model. The mAP@R is a partial AP,

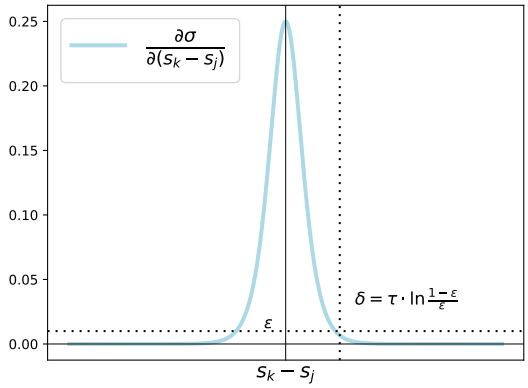


Figure 12: Gradient of the temperature scaled sigmoid ($\tau = 0.01$) *vs* the difference of scores $s_k - s_j$ of a negative pair.

computed on the R first instances retrieved, with R being set to the number of positive instances wrt. a query. mAP@R is a lower bound of the AP (mAP@R = AP when the correct ranking is achieved, *i.e.* mAP@R = AP = 1).

$$mAP@R_i = \frac{1}{R}\sum_{j=1}^{R} P(j), \quad \text{where } P(j) = \begin{cases} \text{precision at j if the jth retrieval is correct} \\ 0 \quad \text{otherwise} \end{cases} \tag{11}$$

B.2 Detail on experimental setup

In this section, we describe the experimental setup used in the Sec. 4.1 of the main paper, and the Sec. B of the supplementary.

We use standard data augmentation strategy during training: images are resized so that their shorter side has a size of 256, we then make a random crop that has a size between 40 and 256, and aspect ratio between 3/4 and 4/3. This crop is then resized to 224x224, and flipped horizontally with a 50% chance. During evaluation, images are resized to 256 and then center cropped to 224.

We use two different strategy to sample each mini-batch. On CUB and INaturalist we choose a batch size (*e.g.* 128) and a number of samples per classes (*e.g.* 4). We then randomly sample classes (*e.g.* 32) to construct our batches. For SOP we use the hard sampling strategy from [3]. For each pair of category (*e.g.* bikes and coffee makers) we use the preceding sampling strategy. This sampling techniques is used because it yields harder and more informative batches. The intuition behind this sampling is that it will be harder to discriminate two bikes from one another, than a bike and a sofa.

We train the ResNet-50 models using Adam [19]. On CUB we train our models with a learning rate of 10^{-6} for 200 epochs. For SOP and INaturalist we take the same scheduling as in [2]. We set the learning rate for the backbone to 10^{-5} and the double for the added linear projection layer. We drop the learning rate by 70% on the epochs 30 and 70. Finally the models are trained for 100 epochs on SOP and 90 on INaturalist (as in [2]).

We train the DeiT transformers models using AdamW [21] as in [7]. On INaturalist we use the same schedule as when training ResNet-50, with a learning rate of 10^{-5}. On SOP we train for 75 epochs with a learning rate of 10^{-5} which is dropped by 70% at epochs 25 and 50. Finally on CUB we train the models for about 100 epochs with a learning rate of 10^{-6}.

B.3 Details of the backbones used

We briefly describe the backbones used throughout out the experiments presented in the main paper and the supplementary.

ResNet-50 [14] We use the well-known convolutional neural network ResNet-50. We remove the linear classification layer. We also add a linear projection layer to reduce the dimension (*e.g.* from 2048 to 512).

DeiT [39] Recently transformer models have been introduced for computer vision [5, 39]. They establish new state-of-the-art performances on computer vision tasks. We use the DeiT-S from [39] which has less parameters than the ResNet-50 ($\sim$ 21 million for DeiT *vs* 25 for ResNet-50). We use the pretrained version with distillation from [39] and its implementation in the `timm` library [45].

B.4 ROADMAP validation

Comparison to AP approximations We compare in Table 5 ROADMAP *vs* other ranking losses on different settings : a batch size of 128 and two backbones (ResNet-50 and DeiT). We conduct this comparison on 5 runs to show the statistical improvement of our method compared to other ranking losses baselines.

We observe that our method outperforms recent ranking losses on the two backbones and the three datasets. On SOP and CUB, ROADMAP has a high increase for the mAP@R, of +1pt on CUB and +2pt on SOP. The performance improvement is greater on the large scale dataset INaturalist with $\sim$+3.5pt with a ResNet-50 backbone and $\sim$+2pt with a DeiT backbone of mAP@R. This trend is the same as in the comparison of the main paper (Table 1).

Table 5: Comparison between ROADMAP and state-of-the-art AP ranking based losses on three image retrieval datasets. *Bck* in the first column stands for bakcbone. Models are trained with a batch size of 128.

		CUB		SOP		INaturalist	
Bck	Method	R@1	mAP@R	R@1	mAP@R	R@1	mAP@R
ResNet-50	FastAP [3]	61.28±0.37	24.11±0.16	78.97±0.05	52.23±0.09	57.23±0.05	22.17±0.05
	SoftBinAP [32]	61.70±0.10	24.29±0.16	80.30±0.21	53.69±0.27	60.88±0.06	23.22±0.05
	BlackBoxAP [33]	61.96±0.28	23.83±0.14	80.97±0.07	54.49±0.15	59.53±0.12	19.62±0.02
	SmoothAP [2]	62.45±0.48	24.32±0.1	81.13±0.05	54.74±0.16	64.48±0.05	24.33±0.07
	ROADMAP (ours)	**64.05**±0.51	**25.27**±0.12	**82.20**± 0.09	**56.64**±0.09	**68.15**±0.10	**27.01**±0.10
DeiT	FastAP [3]	73.42±0.22	31.96±0.06	82.92±0.07	59.06±0.03	62.18±0.07	25.48±0.10
	SoftBinAP [32]	74.84±0.11	33.57±0.08	84.09±0.05	60.53±0.07	65.97±0.13	27.57±0.09
	BlackBoxAP [33]	75.45±0.22	33.97±0.10	84.07±0.09	60.20±0.05	70.29±0.10	29.44±0.06
	SmoothAP [2]	76.02±0.14	34.69±0.08	84.28±0.06	60.49±0.17	69.80±0.08	29.56±0.04
	ROADMAP (ours)	**77.14**±0.12	**36.30**±0.08	**85.44**± 0.06	**62.73**±0.06	**72.81**±0.11	**31.31**±0.10

We perform a paired student t-test to further asses the statistical significance of the performance boost obtained with ROADMAP. We compute the p-values for both the R@1 and mAP@R: it turns out that the p-values are never larger than 0.001, meaning that the gain is statistically significant (with a risk less than 0.1%).

Ablation studies In Table 6 we extend the ablation studies of the main paper (Table 2 of main paper) to other settings, including more batch sizes (32, 128, 224, 384) and two backbones (ResNet-50 and DeiT). On all settings $\mathcal{L}_{\text{SupAP}}$ outperforms the $\mathcal{L}_{\text{SmoothAP}}$ baseline by almost $\sim$+0.5pt consistently, and almost +1pt on every setting for INaturalist. When we add $\mathcal{L}_{\text{calibr.}}$ the gain is further increased. As noticed in Table 2 (main paper) the gain when adding $\mathcal{L}_{\text{calibr.}}$ is particularly noticeable on the large scale dataset INaturalist with boost in performances that can be up to +3.3pt of mAP@R for the ResNet-50 with a batch size 32.

In Table 7 we extend ablation studies with a transformer backbone (DeiT). We observe the same trend as in Table 6. $\mathcal{L}_{\text{SupAP}}$ is consistently better than the $\mathcal{L}_{\text{SmoothAP}}$ baseline, with gain up to more than 1pt (*e.g.* on batch size 128 on INaturalist). $\mathcal{L}_{\text{calibr.}}$ further lifts the performances on the three datasets and all batch sizes.

Table 6: Ablation study for the impact of our two contribution *vs* the SmoothAP baseline for the three datasets and different batch sizes, with a ResNet-50 backbone [14]

				CUB		SOP		INaturalist	
BS	Method	H^-	$\mathcal{L}_{calibr.}$	R@1	mAP@R	R@1	mAP@R	R@1	mAP@R
32	SmoothAP	✗	✗	61.84	23.76	79.96	53.21	53.25	16.4
	SupAP	✓	✗	62.58	24.12	80.51	53.85	55.01	17.13
	ROADMAP	✓	✓	**63.69**	**24.97**	**80.74**	**54.68**	**56.43**	**20.43**
128	SmoothAP	✗	✗	62.81	24.44	81.19	54.96	64.53	24.26
	SupAP	✓	✗	63.18	24.9	81.72	55.65	65.79	24.77
	ROADMAP	✓	✓	**64.18**	**25.38**	**82.18**	**56.64**	**68.28**	**27.13**
224	SmoothAP	✗	✗	62.93	24.69	81.2	54.73	66.62	26.08
	SupAP	✓	✗	64.08	25.13	81.88	55.75	67.43	26.32
	ROADMAP	✓	✓	**64.65**	**25.51**	**82.3**	**56.55**	**69.28**	**27.74**
384	SmoothAP	✗	✗	63.69	24.89	81.45	55.1	67.39	26.77
	SupAP	✓	✗	64.64	25.27	81.94	55.78	68.37	27.24
	ROADMAP	✓	✓	**64.69**	**25.36**	**82.31**	**56.47**	**69.19**	**27.85**

Table 7: Ablation study for the impact of our two contribution *vs* the SmoothAP baseline for the three datasets and different batch sizes, with a DeiT backbone [39]

				CUB		SOP		INaturalist	
BS	Method	H^-	$\mathcal{L}_{calibr.}$	R@1	mAP@R	R@1	mAP@R	R@1	mAP@R
128	SmoothAP	✗	✗	76.2	34.7	84.16	60.18	69.83	29.49
	SupAP	✓	✗	76.33	34.91	84.74	61.29	71.12	30.5
	ROADMAP	✓	✓	**77.09**	**35.76**	**85.44**	**62.57**	**72.82**	**31.36**
224	SmoothAP	✗	✗	76.38	35.33	84.3	60.49	70.55	30.25
	SupAP	✓	✗	76.47	35.67	84.77	61.38	71.9	31.31
	ROADMAP	✓	✓	**77.14**	**36.18**	**85.56**	**62.75**	**73.64**	**31.82**
384	SmoothAP	✗	✗	76.72	35.86	84.66	61.26	71.09	30.89
	SupAP	✓	✗	77.13	36.17	85.01	61.76	72.55	31.89
	ROADMAP	✓	✓	**77.38**	**36.23**	**85.35**	**62.29**	**73.64**	**32.12**

Comparison to state of the art method We show in Table 8 the impact of increasing the embedding dimension when using ResNet-50. All metrics improve on the three datasets when the embedding dimension increases. We observe a gain particularly important on CUB and SOP with $\sim$+1pt in R@1 and mAP@R.

Choosing an embedding size of 2048 further boost the performances of ROADMAP, yielding competitive performances on CUB and state-of-the-art performances for SOP and INaturalist.

Table 8: Difference in performance when using an embedding size of 512 *vs* 2048 with a ResNet-50 backbone, on the three datasets. Performances are obtained with the same setup as described in the Sec. 4.2 of the main paper.

		CUB		SOP		INaturalist	
Method	dim	R@1	mAP@R	R@1	mAP@R	R@1	mAP@R
ROADMAP (ours)	512	68.5	27.97	83.19	58.05	69.19	27.85
ROADMAP (ours)	2048	**69.87**	**28.8**	**83.77**	**59.38**	**69.62**	**27.87**

Preliminary results on Landmarks retrieval We show in Table 9 preliminary experiments to evaluate ROADMAP on $\mathcal{R}$Oxford and $\mathcal{R}$Paris [30], by training our model on the SfM-120k dataset and using the standard GitHub code for evaluation[2].

We can see that ROADMAP is significantly better than [7] with the DeiT-S [39] on $\mathcal{R}$Oxford and $\mathcal{R}$Paris medium protocol, and has similar performances for $\mathcal{R}$Paris hard protocol. This highlights the relevance of using ROADMAP instead of the contrastive loss used in [7].

Table 9: Comparison of ROADMAP vs IRT [7] on $\mathcal{R}$Oxford and $\mathcal{R}$Paris [30]. Models are DeiT-S [39], ROADMAP is trained with a batch size of 128.

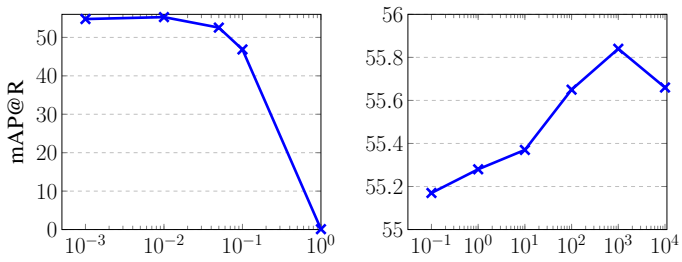
Method	$\mathcal{R}$Oxford		$\mathcal{R}$Paris	
	Medium	Hard	Medium	Hard
IRT [7]	34.5	15.8	65.8	42.0
ROADMAP (ours)	**38.9**	**20.7**	**67.5**	**42.3**

B.5 Model analysis

Hyperparameters In Fig. 13 we show the impact of the hyperparameters of $\mathcal{L}_{\text{SupAP}}$. We plot the mAP@R *vs* τ in Fig. 13a and mAP@R *vs* ρ in Fig. 13b. The experiments are conducted on SOP with a batch size of 128.

We observe on Fig. 13a that $\mathcal{L}_{\text{SupAP}}$ is stable with small values of τ, *i.e.* in the range [0.001, 0.05]. As a reminder we use the default value $\tau = 0.01$ in all our results, as it was the suggested value from the SmoothAP paper [2].

We conduct a study of the impact of ρ in Fig. 13b. We find that $\mathcal{L}_{\text{SupAP}}$ is very stable wrt. this hyperparameter. Performances are improving with a greater value of ρ before dropping after 10^4. The trend follows what was observed in the Fig. 4b of the main paper, although this time using a value if $\rho = 10^4$ yields better performances. Using cross-validation to choose an optimal value for ρ may lead to even better performances for $\mathcal{L}_{\text{SupAP}}$.



(a) mAP@R *vs* τ for $\mathcal{L}_{\text{SupAP}}$. (b) mAP@R *vs* ρ for $\mathcal{L}_{\text{SupAP}}$.

Figure 13: Analysis of $\mathcal{L}_{\text{SupAP}}$ hyperparameters on SOP (batch size 128).

Decomposability gap In Table 10 we measure the relative decrease of the decomposability gap DG_{AP} on SOP and CUB test sets. On both datasets we can see that $\mathcal{L}_{\text{calibr.}}$ decreases the decomposability gap.

Table 10: Relative decrease of the decomposability gap when adding $\mathcal{L}_{\text{calibr.}}$ to $\mathcal{L}_{\text{SupAP}}$ (ROADMAP).

Dataset	decrease of DG_{AP}
CUB	3.7%
SOP	5.4%

[2] `https://github.com/filipradenovic/cnnimageretrieval-pytorch`

B.6 Source code

We describe in this section the software used for our work, and discuss the computation costs associated with training models presented in this paper.

Librairies We use several Python libraries often used in image retrieval.

We use `PyTorch` [29] as a general framework to implement our neural networks, losses and training loops. We use several utilities from `PyTorch Metric Learing` [27], an open-source Python library focused on helping researcher working on image retrieval and metric learning. We use `Faiss` [18] to compute metrics (*i.e.* to perform nearest neighbours search), which is a Python library often used in image retrieval to compute the rankings or the similarity matrix. To load and use the transformer models we use `timm` [45], a library implementing recent computer vision models, with pretrained weights for most of them. To handle all our config files, we use `Hydra` [49], this library makes it possible to combine the use of Yaml configuration files and overriding them using the command line.

We use the publicly available implementation of SoftBinAP[3] [32] which is under a BSD-3 license. The original codes of SmoothAP[4] [2], BlackBox[5] [28, 33] are under an MIT license. For FastAP [3] we use the implementation from [27] (MIT license), the original implementation of FastAP[6] is also under an MIT license.

Compute costs We use mixed-precision learning offered within PyTorch [29]. The time and memory consumption are reduced by a factor between 2 and $3/2$ with no notable difference in performances. We could train all models on 16GiB GPUs, except for models trained with a batch size of 384 which requires a 32GiB GPU.

CUB Models take between 30 minutes and 1 hour to train on a Nvidia Quadro RTX 5000 with 16GiB.

SOP Models take between 4 and 8 hours to train on a Nvidia Quadro RTX 5000 with 16GiB.

INaturalist To train models on INaturalist we were granted access to the IDRIS HPC cluster with Tesla V-100 GPUs (of 16GiB or 32GiB). Models train for approximately 20 hours.

We could not train models with mixed-precision when using BlackBox [33]. Models trained with it took longer to train (*e.g.* 30 hours on INaturalist) and are more demanding on memory (almost 16GiB with a batch size of 128 while models trained with other loss functions required less than 10Gib).

C Qualitative results

CUB As a qualitative assessment, we show in Fig. 14 some results of ROADMAP on CUB. We show the queries (in purple) and the 10 most similar retrieved images, with relevant instances in green and irrelevant instances in red.

SOP In Fig. 15 we perform the same assessment for SOP. In SOP there are fewer relevant instances per query (in average 5). So even for queries that retrieved all the relevant instances, there will be negative instances that have high ranks (in Fig. 15 ranks that are lower than 10).

INaturalist Finally we show on Fig. 16 some examples of queries and the 10 most similar instances for a model trained with ROADMAP on INaturalist.

[3] `https://github.com/naver/deep-image-retrieval`

[4] `https://github.com/Andrew-Brown1/Smooth_AP`

[5] `https://github.com/martius-lab/blackbox-backprop`

[6] `https://github.com/kunhe/FastAP-metric-learning`

Figure 14: Qualitative results on CUB: a query (purple) with the 10 most similar instances. Relevant (resp. irrelevant) instances are in green (resp. red).

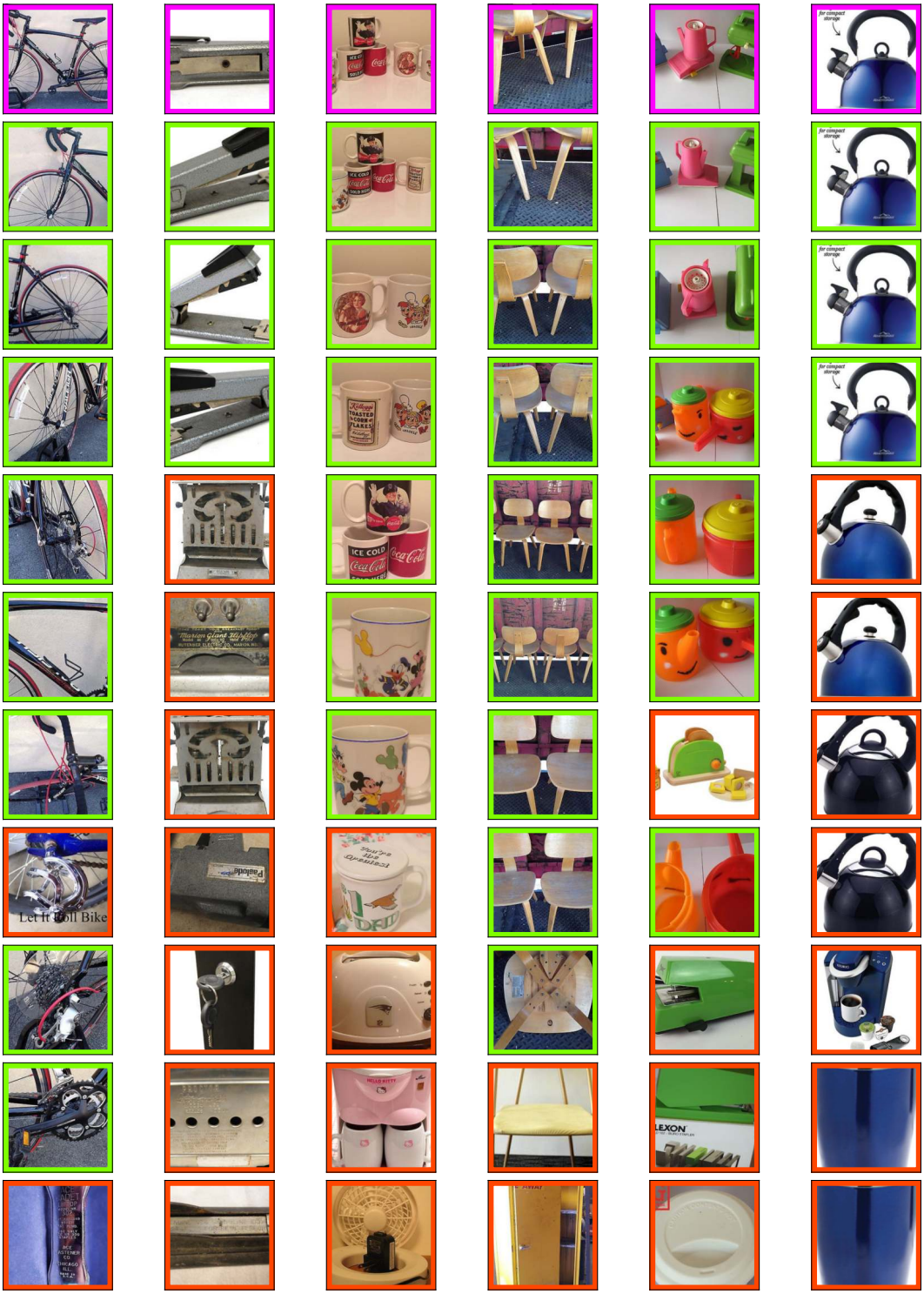


Figure 15: Qualitative results on SOP: a query (purple) with the 10 most similar instances. Relevant (resp. irrelevant) instances are in green (resp. red).

Figure 16: Qualitative results on INaturalist: a query (purple) with the 10 most similar instances. Relevant (resp. irrelevant) instances are in green (resp. red).