Optimization of Rank Losses for Image Retrieval

Elias Ramzi, Nicolas Audebert, Clément Rambour,
André Araujo , *Senior Member, IEEE,* Xavier Bitot, and Nicolas Thome

***Abstract*—In image retrieval, standard evaluation metrics rely on score ranking, *e.g.* average precision (AP), recall at k (R@k), normalized discounted cumulative gain (NDCG). In this work, we introduce a general framework for robust and decomposable rank losses optimization. It addresses two major challenges for end-to-end training of deep neural networks with rank losses: non-differentiability and non-decomposability. Firstly, we propose a general surrogate for ranking operator, SupRank, that is amenable to stochastic gradient descent. It provides an upperbound for rank losses and ensures robust training. Secondly, we use a simple yet effective loss function to reduce the decomposability gap between the averaged batch approximation of ranking losses and their values on the whole training set. We apply our framework to two standard metrics for image retrieval: AP and R@k. Additionally, we apply our framework to hierarchical image retrieval. We introduce an extension of AP, the hierarchical average precision $\mathcal{H}$-AP, and optimize it as well as the NDCG. Finally, we create the first hierarchical landmarks retrieval dataset. We use a semi-automatic pipeline to create hierarchical labels, extending the large scale Google Landmarks v2 dataset. The hierarchical dataset is publicly available at github.com/cvdfoundation/google-landmark. Code is available at github.com/elias-ramzi/SupRank.**

***Index Terms*—Image Retrieval, Ranking, Average Precision, Hierarchical Ranking, Hierarchical Average Precision, Non-Decomposable**

I. Introduction

Image retrieval (IR) is a major task in computer vision. Its goal is to retrieve "similar" images to a query in a database. In modern computer vision this is achieved by learning a space of image representation, *i.e.* embeddings, where "similar" images are close to each other.

The performances of IR systems are often measured using ranking-based metrics, *e.g.* average precision (AP), recall rate at k (R@k), Normalized Discounted Cumulative Gain (NDCG). These metrics penalize assigning a higher similarity to non-relevant images than to relevant ones.

Although these metrics are well-suited for image retrieval, their use as a loss functions for training deep neural networks is a challenge. We can point out two two main difficulties: i) they are not amenable to stochastic gradient descent (SGD) and thus cannot be used directly to train deep neural networks (DNN), ii) they are not decomposable.

Regarding the first challenge, there has been a rich literature to provide proxy losses for the task of image retrieval using tuplet losses [1]–[9] or cross entropy based losses [10]–[15]. There also has been extensive work to create rank losses amenable to gradient descent [16]–[28]. They create either coarse upper bounds of the target metric or tighter approximations but loosen the upper bound property which affects final performances.

Elias Ramzi, Nicolas Audebert and Clément Rambour are with the Cnam.
André Araujo is with Google DeepMind.
Xavier Bitot is with Coexya.
Nicolas Thome is with Sorbonne Université.
Manuscript updated August 2nd, 2024

The non-decomposability challenge arises when training deep models with ranking losses, since the average loss over batches generally underestimates its value on the whole training dataset, which we refer to as the *decomposability gap*. In image retrieval, attempts to circumvent the problem involve *ad hoc* methods based on hard batch sampling strategies [5], [7], [29], [30], storing all training representations/scores [31], [32] or using larger batches [24], [25], [28], leading to complex models with a large computation or memory overhead.

The core of our approach is a a unified framework, illustrated in Fig. 1 and detailed in Sec. III, to optimize rank losses for both hierarchical and standard image retrieval. Specifically, we propose SupRank, a smooth approximation of the rank which is amenable to SGD and is an upper bound on the true rank. SupRank leads to smooth losses that are upper bounds of the true losses. At training time, we additionally introduce a novel objective to reduce the non-decomposability of smooth rank losses without the need to increase the batch size.

Our framework for end-to-end training of DNN is illustrated in Fig. 1. Using a DNN f_θ we encode both the query and the other images of the batch. Optimizing the rank loss supports the correct–partial–ordering in a batch based on our surrogate of the rank, SupRank. Optimizing the decomposability loss supports that the positives will be ranked before negative items , even those that are not present in the batch. Both losses are amenable to gradient descent, which makes possible to update the model parameters with SGD.

Our framework can be used to optimize rank losses for both hierarchical and non-hierarchical image retrieval. In a first time we show how to instantiate our framework to non-hierarchical image retrieval by optimizing two ranking-based metrics, namely AP and R@k. We show the importance of the two components of our framework in ablation studies. Using our AP surrogate, we achieve state-of-the-art image retrieval performances across 3 datasets and 3 neural networks architectures.

In a second instantiation we focus on hierarchical image retrieval [33]–[35]. Because metrics used to evaluate fine-grained image retrieval rely on binary labels, *i.e.* similar or dissimilar, they are unable to take into account the severity of the errors. This leads methods that optimize this metrics to lack robustness: they tend to make severe errors when they make some. Hierarchical image retrieval can be used to mitigate this issue by taking into account non-binary similarity between labels. We introduce the hierarchical average precision, $\mathcal{H}$-AP, a new metric that extends the AP to non-binary settings. Using our optimization framework, we exhibit how optimizing the $\mathcal{H}$-AP and the well known NDCG leads to competitive results for fine-grained image retrieval metrics, while outperforming by large margins both binary methods and hierarchical baselines when considering hierarchical metrics.

Finally we introduce the first hierarchical landmarks retrieval dataset, $\mathcal{H}$-GLDv2, extending the well-known Google Landmarks

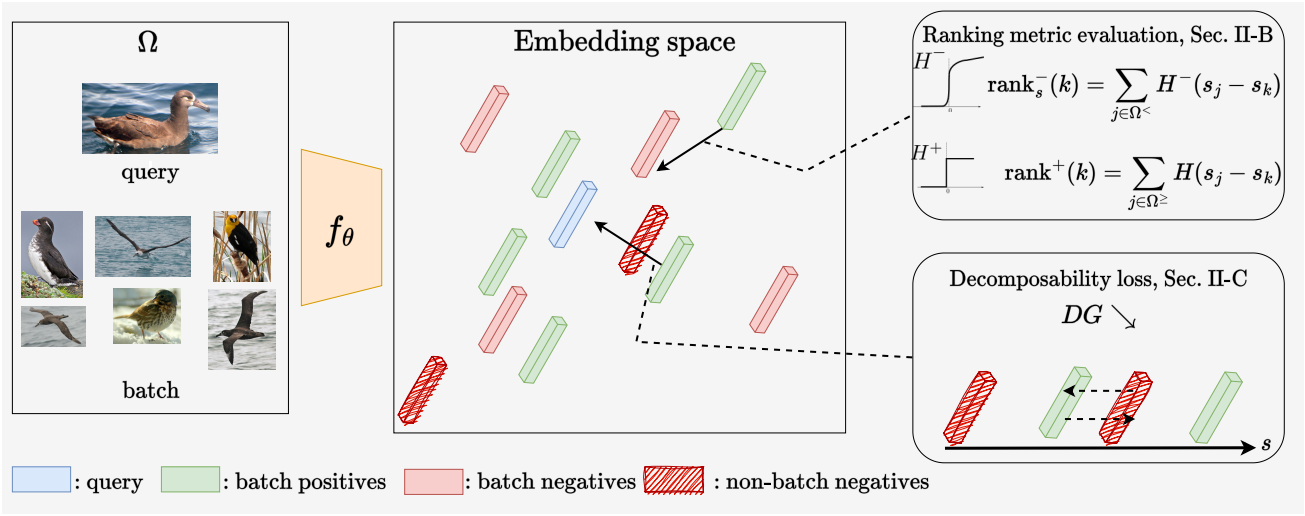


Fig. 1: Illustration of our unified framework which supports both hierarchical and non-hierarchical cases. We use a deep neural network f_θ to embed images. We then optimize its weights in an end-to-end manner using two losses: 1) we optimize the ranking-based evaluation metric using an upper bound approximation of the rank, rank_s^-, as described in Sec. III-B, enforcing the batch's positive embeddings to have higher cosine similarity with the query than the batch's negatives; 2) we reduce the decomposability gap, DG, of rank losses using a decomposability loss as described in Sec. III-C, that supports that positives have higher similarity with the query than all negatives even outside the batch.

v2 landmarks retrieval (GLDv2) dataset [36]. While landmarks retrieval has been one of the most popular domain in image retrieval it lacks a hierarchical dataset. $\mathcal{H}$-GLDv2 is a large scale dataset with 1.4m images and three levels of hierarchies: including 100k unique landmarks, 78 super-categories and 2 final labels. The labels are publicly available at github.com/cvdfoundation/google-landmark.

Initial results of our work have been presented in [35], [37]. In this work, we unify the methods from these two papers into a framework for the optimization of rank losses, naturally supporting both standard and hierarchical image retrieval problems. Additionally, we include more comprehensive experiments, to consider different decomposability objectives, apply our framework to the recent R@k loss [28] and optimize the NDCG in the hierarchical setting. Finally, in this work we introduce the first hierarchical image retrieval dataset in the domain of landmarks, which is incorporated for a more comprehensive benchmarking of our method.

II. Related work

A. Image Retrieval proxy losses

The Image Retrieval community has designed several families of methods to optimize metrics such as AP and R@k. Methods that rely on tuplet-wise losses, like pair losses [1]–[3], triplet losses [4]–[6], or larger tuplets [7]–[9] learn comparison relations between instances. These metric learning methods optimize a very coarse upper bound on AP and need complex post-processing and tricks to be effective. Other methods using proxies have been introduced to lower the computational complexity of tuplet based training [10]–[15]: they learn jointly a deep model and weight matrix that represent proxies using a cross-entropy based loss. Proxies are approximations of the original data points that should belong to their neighborhood.

B. Rank loss approximations

Studying smooth rank surrogate losses has a long history. One option for training with rank losses is to design smooth upper bounds. Seminal works are based on structural SVMs [16], [17], with extensions to speed-up the "loss-augmented inference" [18] or to adapt to weak supervision [19] were designed to optimize AP. Generic black-box combinatorial solvers have been introduced [20] and applied to AP optimization [32]. 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 [21]–[25] by using a smoothed discretization of similarity scores. Other approaches rely on explicitly approximating the non-differentiable rank functions using neural networks [26], or with a sum of sigmoid functions in the Smooth-AP approach [27] or the more recent Smooth-Recall loss [28]. These approaches enable accurate surrogates by providing tight and smooth approximations of the rank function. However, they do not guarantee that the resulting loss is an upper bound on the true loss. The SupRank introduced in this work is based on a smooth approximation of the rank function leading to an upper bound on the true loss, making our approach both accurate and robust.

C. 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 [29], [30], [38] or selecting informative constraints within mini-batches [7], [30], [39], [40]. In cross-batch memory technique [31], 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 [20], [24], [25], [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 does not add any overhead and enables good performances for AP optimization even with small batches.

D. Hierarchical predictions and metrics

There has been a recent regain of interest in Hierarchical Classification (HC) [41]–[43], to learn robust models that make "better mistakes" [42]. However HC is evaluted in *closed set*, *i.e.* train and test classes are the same. Whereas, hierarchical image retrieval considers the *open set* paradigm, where classes are distinct between train and test sets to better evaluate the generalization abilities of learned models.

The Information Retrieval community uses datasets where documents can be more or less relevant depending on the query [44], [45]. The quality of their retrieval engine is quantified using ranking based metrics such as the NDCG [46], [47]. Several works have investigated how to optimize the NDCG, *e.g.* using pairwise losses [48] or smooth surrogates [49]–[52]. These works however focused on NDCG, and are without any theoretical guarantees: the surrogates are approximations of the NDCG but not *lower bounds*, *i.e.* their maximization does not imply improved performances during inference. An additional drawback is that NDCG does not relate easily to average precision [53], the most common metric in image retrieval. Fortunately, there have been some works done to extend AP in a graded setting where relevance between instances is not binary [54], [55]. The graded Average Precision from [54] is the closest to our work as it leverages SoftRank for direct optimization of non-binary relevance, although there are significant shortcomings. There is no guarantee that the SoftRank surrogate actually minimizes the graded AP, it requires to annotate datasets with pairwise relevances which is impractical for large scale settings in image retrieval.

Recently, the authors of [33] introduced three new hierarchical benchmarks datasets for image retrieval, in addition to a novel hierarchical loss CSL. CSL extends proxy-based triplet losses to the hierarchical setting. However, this method faces the same limitation as triplet losses: minimizing CSL does not explicitly optimize a well-behaved hierarchical evaluation metric, *e.g.* $\mathcal{H}$-AP. We show experimentally that our method significantly outperforms CSL [33] both on hierarchical metrics and AP-level evaluations.

E. Hierarchical datasets

Hierarchical trees are available for a large number of datasets, such as CUB-200-2011 [56], Cars196 [57], InShop [58], Stanford Online Products [59], and notably *large-scale* ones such as iNaturalist [60], the three DyML datasets [33] and Imagenet [61]. Hierarchical labels are also less difficult to obtain than fine-grained ones since hierarchical relations can be semi-automatically obtained by grouping fine-grained labels. This was previously done by [43] or by using the large lexical database Wordnet [62] *e.g.* for Imagenet in [61] and for the SUN database in [63]. In the same spirit, we introduce for the first time a hierarchical dataset for the landmark instance retrieval problem: $\mathcal{H}$-GLDv2. We extend the well-known Google Landmarks Dataset v2 [36] with hierarchical labels using a semi-automatic pipeline, leveraging category labels mined from Wikimedia commons and substantial manual cleaning.

III. Smooth and decomposable rank losses

A. Preliminaries

Let us consider a retrieval set $\Omega = \{\boldsymbol{x_j}\}_{j\in[\![1;N]\!]}$ composed of N elements, and a set of M queries $\mathcal{Q}$. For each query $\boldsymbol{q_i}$, each element in Ω is assigned a relevance $\mathrm{rel}(\boldsymbol{x_j},\boldsymbol{q_i}) \in \mathbb{R}$ [44], such that $\mathrm{rel}(\boldsymbol{x_j},\boldsymbol{q_i}) > 0$ (resp. $\mathrm{rel}(\boldsymbol{x_j},\boldsymbol{q_i}) = 0$) if $\boldsymbol{x_j}$ is relevant (resp. irrelevant) with respect to $\boldsymbol{q_i}$. For the standard image retrieval discussed in Sec. IV, $\mathrm{rel}(\boldsymbol{x_j},\boldsymbol{q_i}) = 1$ if x_j and q_i share the same fine-grained label and 0 otherwise. In the hierarchical image retrieval setting $\mathrm{rel}(\boldsymbol{x_j},\boldsymbol{q_i})$ models more complex pairwise relevance discussed in Sec. V. A positive relevance defines the set of positives for a query, *i.e.* $\Omega_i^+ := \{\boldsymbol{x_j} \in \Omega | \mathrm{rel}(\boldsymbol{x_j},\boldsymbol{q_i}) > 0\}$, *i.e.* a positive instance in standard image retrieval or an instance with relevance greater than 0 in the hierarchical case. Instances with a relevance of 0 are the negatives, *i.e.* $\Omega_i^- := \{\boldsymbol{x_j} \in \Omega | \mathrm{rel}(\boldsymbol{x_j},\boldsymbol{q_i}) = 0\}$.

For each $\boldsymbol{x_j} \in \Omega$, we compute its embedding $\mathbf{v_j} \in \mathbb{R}^d$. To do so we use a neural network $f_{\boldsymbol{\theta}}$ parameterized by $\boldsymbol{\theta}$: $\mathbf{v_j} := f_{\boldsymbol{\theta}}(\boldsymbol{x_j})$. In the embedding space $\mathbb{R}^d$, we compute the cosine similarity score between each query $\boldsymbol{q_i}$ and each element in Ω: $s(\boldsymbol{q_i},\boldsymbol{x_j}) = \mathbf{v_{q_i}}^T \mathbf{v_j} / ||\mathbf{v_{q_i}}|| \cdot ||\mathbf{v_j}||$.

During training, our goal is to optimize, for each query $\boldsymbol{q_i}$, the model parameters $\boldsymbol{\theta}$ such that the ranking, *i.e.* decreasing order of cosine similarity, matches the ground truth ranking, *i.e.* decreasing order of relevances. More precisely, we optimize a ranking-based metric $0 \leq \mathcal{M}_i \leq 1$ that penalizes inversion between positive instances and negative ones. The target loss is averaged over all queries:

$$\mathcal{L}_{\mathcal{M}}(\boldsymbol{\theta}) = 1 - \frac{1}{M}\sum_{i=1}^{M} \mathcal{M}_i(\boldsymbol{\theta}) \tag{1}$$

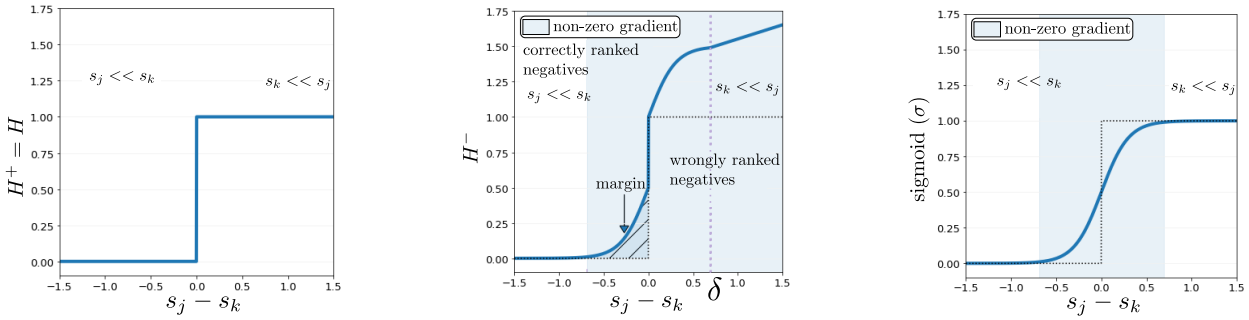
As previously mentioned, there are two main challenges with SGD optimization of rank losses: i) they are not differentiable with respect to $\boldsymbol{\theta}$, and ii) they do not linearly decompose into batches. We propose to address both issues: we introduce a robust differentiable ranking surrogate, SupRank (Sec. III-B), and add a decomposable objective (Sec. III-C) to improve rank losses' behavior in a batch setting. Our final **RO**bust and **D**ecomposable (ROD) loss $\mathcal{L}_{\text{ROD-}\mathcal{M}}$ combines a differentiable surrogate loss of a target ranking-based metric, $\mathcal{L}_{\text{Sup-}\mathcal{M}}$, and the decomposable objective $\mathcal{L}_{\text{DG}}$ with a linear combination, weighted by the hyper-parameter λ:

$$\mathcal{L}_{\text{ROD-}\mathcal{M}}(\boldsymbol{\theta}) = (1-\lambda)\cdot\mathcal{L}_{\text{Sup-}\mathcal{M}}(\boldsymbol{\theta}) + \lambda\cdot\mathcal{L}_{\text{DG}}(\boldsymbol{\theta}) \tag{2}$$

B. SupRank: smooth approximation of the rank

The non-differentiablity in rank losses comes from the ranking operator, which can be viewed as counting the number of instances that have a similarity s_j greater than the considered instance x_k—for readability we drop the dependence on $\boldsymbol{\theta}$ and on the query q_i—, *i.e.*:

$$\mathrm{rank}(k) = 1 + \underbrace{\sum_{j\in\Omega_k^{\geq}} H(s_j - s_k)}_{\mathrm{rank}^+(k)} + \underbrace{\sum_{j\in\Omega_k^{<}} H(s_j - s_k)}_{\mathrm{rank}^-(k)} \tag{3}$$



(a) $H^+(t) = H(t)$ in Eq. (3) (b) $H^-(t)$ in Eq. (4) (c) Sigmoid used in [27]

Fig. 2: Proposed surrogate losses for the Heaviside (step), for $t = s_j - s_k$: with H^+ in Fig. 2a and H^- in Fig. 2b. Using H^- in Eq. (5) leads to smooth and upperbounds rank losses. In addition, H^- back-propagates gradients until the correct ranking is satisfied, in contrast to the sigmoid used in [27] (Fig. 2c).

where H is the Heaviside (step) function $H(t) = 1$ if $t \geq 0$, 0 otherwise; $\Omega_k^{\geq} = \{\boldsymbol{x_p} \in \Omega | \text{rel}(\boldsymbol{x_p}, \boldsymbol{q}) \geq \text{rel}(\boldsymbol{x_k}, \boldsymbol{q})\}$, *i.e.* the set of instances with a relevance greater or equal to k's, and $\Omega_k^{<} = \{\boldsymbol{x_p} \in \Omega | \text{rel}(\boldsymbol{x_p}, \boldsymbol{q}) < \text{rel}(\boldsymbol{x_k}, \boldsymbol{q})\}$ the set of instances with a relevance strictly lower to k's (in standard IR $\Omega_k^{<} = \Omega^-$). Note that for both $\text{rank}^+(k)$ and $\text{rank}^-(k)$ in Eq. (3) k is always positive, *i.e.* in Ω^+, and x_j can either be negative, *i.e.* in Ω^-, in rank^- or positive in rank^+, *i.e.* in Ω^+.

From Eq. (3) it becomes clear that the rank is non-amenable to gradient descent optimization due to the Heaviside (step) function H (see Fig. 2a), whose derivatives are either zero or undefined.

SupRank. To provide rank losses amenable to SGD, we introduce a smooth approximation of the rank function. We propose a different behavior between $\text{rank}^+(k)$ and $\text{rank}^-(k)$ in Eq. (3) by defining two functions H^+ and H^-. For $\text{rank}^+(k)$, we keep the Heaviside function, *i.e.* $H^+ = H$ (see Fig. 2a). This ignores $\text{rank}^+(k)$ in gradient-based ranking optimization. It has been observed in other works that optimizing rank^- is sufficient [64]. For $\text{rank}^-(k)$ we want smooth surrogate H^- for H that is a amenable to SGD and an upper bound on the Heaviside function. We define the following H^- function for $t \in \mathbb{R}$, illustrated in Fig 2b, that is both:

$$H^-(t) = \begin{cases} \sigma(\frac{t}{\tau}) & \text{if } t \leq 0 \\ \sigma(\frac{t}{\tau}) + 0.5 & \text{if } t \in [0; \delta] \\ \rho \cdot (t - \delta) + \sigma(\frac{\delta}{\tau}) + 0.5 & \text{if } t > \delta \end{cases} \quad \text{with } \delta \geq 0 \quad (4)$$

where σ is the sigmoid function (Fig. 2c), δ the offset of the linear section, ρ the slope of the linear section, and τ the temperature of the sigmoid that controls the margin on Fig. 2b are hyper-parameters. δ is chosen such that the sigmoidal part of H^- reaches the saturation regime and is fixed for the rest of the paper (see supplementary Sec. A-C). We keep τ as in [27] and study the robustness to ρ in Sec. VII-A4.

From H^- in Eq. (4), we define the following rank surrogate that can be used plug-and-play for rank losses optimization:

$$\text{rank}_s^-(k) = \sum_{j \in \Omega_k^{<}} H^-(s_j - s_k) \quad (5)$$

SupRank has two main features:

► ① **Surrogate losses based on SupRank are upper bound of the target metrics** , since H^- in Eq. (4) is an upper bound of a step function (Fig 2b). This is an 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 [21], [24], [25], [27] do not fulfill this property.

► ② **SupRank brings training gradients until the correct ranking plus a margin is fulfilled.** When the ranking is incorrect, an instance with a lower relevance $\boldsymbol{x_j}$ is ranked before an instance of higher relevance $\boldsymbol{x_k}$, thus $s_j > s_k$ and $H^-(s_j - s_k)$ in Eq. (4) has a non-zero 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 parameterized by τ (sigmoid in Eq. (4)). This margin overcomes the brittleness of rank losses, which vanish as soon as the ranking is correct [20], [22], [24].

C. Decomposable rank losses

As illustrated in Eq. (1), rank losses decompose linearly between queries $\boldsymbol{q_i}$, but do not between retrieved instances. 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 $\mathcal{M}_b(\boldsymbol{\theta})$ be the metric $\mathcal{M}$ in batch b for a query, we define the "decomposability gap" DG as:

$$DG(\boldsymbol{\theta}) = \mathcal{M}(\boldsymbol{\theta}) - \frac{1}{K} \sum_{b=1}^{K} \mathcal{M}_b(\boldsymbol{\theta}) \quad (6)$$

DG in Eq. (6) is a direct measure of the non-decomposability of any metric $\mathcal{M}$ (illustrated for AP in Sec. A-A). Our motivation here is to decrease DG, *i.e.* to have the average metric over the batches as close as possible to the metric computed over the whole training set. To this end, we use a additional objective during training that aims at reducing the non-decomposability.

Pair-based decomposability loss. We use the following decomposability loss $\mathcal{L}_{\text{DG}}$ that was first introduced in ROADMAP [37], and used in other work [65] to reduce the non-decomposability of ranking losses:

$$\mathcal{L}_{\text{DG}}(\boldsymbol{\theta}) = \frac{1}{|\Omega^+|} \sum_{\boldsymbol{x_j} \in \Omega^+} [\alpha - s_j]_+ + \frac{1}{|\Omega^-|} \sum_{\boldsymbol{x_j} \in \Omega^-} [s_j - \beta]_+ \quad (7)$$

where $[x]_+ = \max(0, x)$. $\mathcal{L}_{\text{DG}}$ is a pair-based loss [2], which we revisit in our context to "calibrate" the scores between mini-batches. Intuitively, the fact that the positive (resp. negative) scores are above (resp. below) a threshold α (resp. β) in the mini-batches makes $\mathcal{M}_b$ closer to $\mathcal{M}$, which we support with an analysis in Sec. A-B.

Proxy-based decomposability loss. In HAPPIER [35] we used the following proxy-based loss as the decomposability objective:

$$\mathcal{L}^*_{\text{DG}}(\theta) = -\log\left(\frac{\exp(\frac{v_y^T p_y}{\eta})}{\sum_{p_z\in\mathcal{Z}}\exp(\frac{v_y^T p_z}{\eta})}\right), \tag{8}$$

where p_y is the normalized proxy corresponding to the fine-grained class of the embedding v_y, $\mathcal{Z}$ is the set of proxies, and η is a temperature scaling parameter. $\mathcal{L}^*_{\text{DG}}$ is a classification-based proxy loss [11] that imposes a margin instances and the proxies. $\mathcal{L}^*_{\text{DG}}$ has thus a similar effect to $\mathcal{L}_{\text{DG}}$ on the decomposability of rank losses. In our experiments we show that both decomposability losses improve ranking losses optimization.

IV. Instantiation to standard image retrieval

In this section we apply the framework described previously to standard image retrieval where $\text{rel}(x,q)\in\{0,1\}$. Specifically we show how to directly optimize two metrics that are widely used in the image retrieval community, *i.e.* AP and R@k.

A. Application to Average Precision

The average precision measures the quality of a ranking by penalizing inversion between positives and negatives. It strongly penalizes inversion at the top of the ranking. It is defined for each query q_i as follows:

$$\text{AP}_i = \frac{1}{|\Omega_i^+|}\sum_{k\in\Omega_i^+}\frac{\text{rank}^+(k)}{\text{rank}(k)} \tag{9}$$

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

$$\mathcal{L}_{\text{AP}}(\boldsymbol{\theta}) = 1 - \frac{1}{M}\sum_{i=1}^{M}\text{AP}_i(\boldsymbol{\theta}) \tag{10}$$

Using our surrogate of the rank, SupRank, we define the following AP surrogate loss:

$$\mathcal{L}_{\text{Sup-AP}}(\boldsymbol{\theta}) = 1 - \frac{1}{M}\sum_{i=1}^{M}\frac{1}{|\Omega_i^+|}\sum_{k\in\Omega_i^+}\frac{\text{rank}^+(k)}{\text{rank}^+(k)+\text{rank}^-_s(k)} \tag{11}$$

Finally we equip the AP surrogate loss with the $\mathcal{L}_{\text{DG}}$ loss to support the decomposability of the AP, yielding our **RO**bust **A**nd **D**eco**M**posable **A**verage **P**recision:

$$\mathcal{L}_{\text{ROADMAP}}(\boldsymbol{\theta}) = (1-\lambda)\cdot\mathcal{L}_{\text{Sup-AP}}(\boldsymbol{\theta}) + \lambda\cdot\mathcal{L}_{\text{DG}}(\boldsymbol{\theta}) \tag{12}$$

B. Application to the Recall at k

Another metric often used in image retrieval is the recall rate at k. In the image retrieval community it is often defined as:

$$\text{R@k} = \frac{1}{M}\sum_{i=1}^{M}\mathbb{1}(\text{positive element in top-}k) \tag{13}$$

However in the literature the recall is most often defined as:

$$\text{TR@k} = \frac{1}{M}\sum_{i=1}^{M}\frac{\#\text{ positive elements in top-}k}{\min(k,\#\text{ positive elements})} \tag{14}$$

It was shown in [28] that the TR@k can be written similarly to other ranking-based metrics, *i.e.* using the rank, for each query q_i as:

$$\text{TR@k} = \frac{1}{M}\sum_{i=1}^{M}\frac{1}{\min(|\Omega_i^+|,k)}\sum_{p\in\Omega_i^+}H(k-\text{rank}(p)) \tag{15}$$

Using the expression of Eq. (15) and SupRank we can derive a surrogate loss function for the recall for a single query as:

$$\mathcal{L}_{\text{Sup-R@k}} = 1 - \frac{1}{\min(|\Omega^+|,k)}\sum_{p\in\Omega^+}\sigma\left(\frac{k-(\text{rank}^+(p)+\text{rank}^-_s(p))}{\tau^*}\right) \tag{16}$$

The authors of [28] use different level of recalls in their loss, which we follow *i.e.* $\mathcal{L}_{\text{Sup-R@}\mathcal{K}} = \frac{1}{|\mathcal{K}|}\sum_{k\in\mathcal{K}}\mathcal{L}_{\text{Sup-R@k}}$, it is necessary to provide enough gradient signal to all positive items. To train $\mathcal{L}_{\text{Sup-}R@k}$, it is also necessary to approximate a second time the Heaviside function, using a sigmoid with temperature factor τ^*. We combine it with $\mathcal{L}_{\text{DG}}$ yielding the resulting differentiable and decomposable R@k loss:

$$\mathcal{L}_{\text{ROD-R@}\mathcal{K}} = (1-\lambda)\cdot\mathcal{L}_{\text{Sup-R@}\mathcal{K}} + \lambda\cdot\mathcal{L}_{\text{DG}} \tag{17}$$

V. Instantiation to Hierarchical Image Retrieval

Standard metrics (*e.g.* AP or R@k) are only defined for binary labels, *i.e. fine-grained* labels: an image is negative if it is not strictly similar to the query. These metrics are by design unable to take into account the severity of the mistakes. To mitigate this issue we propose to optimize a new ranking-based metric, $\mathcal{H}$-AP introduced in Sec. V-A, that extends AP beyond binary labels, and the standard NDCG in Sec. V-B.

Additional training context. We assume that we have access to a hierarchical tree defining semantic similarities between concepts as in Fig. 3. For a query $\boldsymbol{q}$, we partition the set of retrieved instances into $L+1$ disjoint subsets $\left\{\Omega^{(l)}\right\}_{l\in[\![0;L]\!]}$. $\Omega^{(L)}$ is the subset of the most similar instances to the query (*i.e.* fine-grained level): *e.g.* on Fig. 3 given the query in purple, $\Omega^{(3)}$ consists of all images of "Lada #2" (green), *i.e.* same fine-grained classes as the query. For $l<L$, $\Omega^{(l)}$ contains instances with smaller relevance with the query. For instance, $\Omega^{(2)}$ in Fig. 3 is the set of "Lada" cars that are not of the same fine-grained class as the query (*i.e.* "Lada #2"), in light blue. We define $\Omega^- := \Omega^{(0)}$ as the set of negative instances, *i.e.* the set of vehicles that are not "Cars", *e.g.* Pickups or Buses (in red). Finally, we define $\Omega^+ = \bigcup_{l=1}^{L}\Omega^{(l)}$. Given a query q, we use this partition to define the relevance of $k\in\Omega^{(l)}$, $\text{rel}(k) := \text{rel}(x_k,q)$.

A. Hierarchical Average Precision

We propose an extension of AP that leverages non-binary labels. To do so, we extend rank^+ to the hierarchical case with a hierarchical rank^+, $\mathcal{H}$-rank^+:

$$\mathcal{H}\text{-rank}^+(k) = \text{rel}(k) + \sum_{j\in\Omega^+}\min(\text{rel}(k),\text{rel}(j))\cdot H(s_j - s_k)\ . \tag{18}$$

Intuitively, $\min(\text{rel}(k),\ \text{rel}(j))$ corresponds to seeking the closest ancestor shared by instance k and j with the query in the hierarchical tree. As illustrated in Fig. 4, $\mathcal{H}$-rank^+ induces a smoother penalization for instances that do not share the same

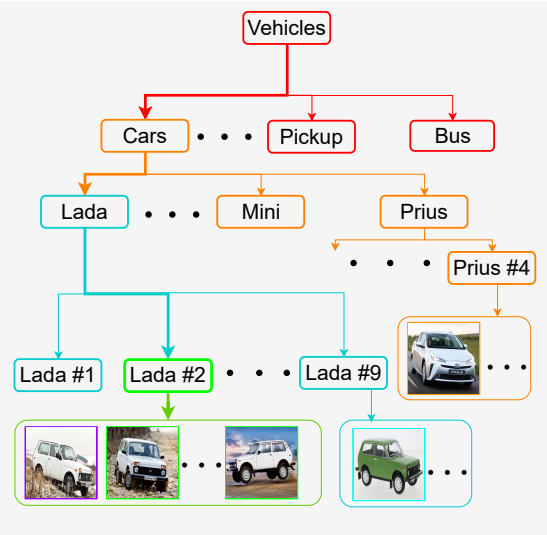


Fig. 3: We leverage a hierarchical tree representing the semantic similarities between concepts to produce more robust ranking.

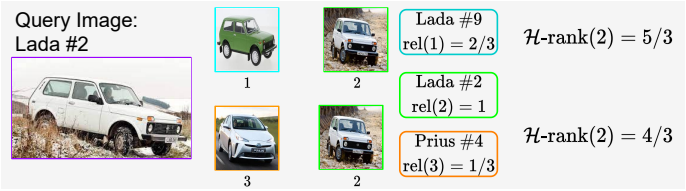


Fig. 4: Given a "Lada #2" query, the top inversion is less severe than the bottom one. Indeed on the top row instance 1 is semantically closer to the query – it is a "Lada"– than instance 3 on the bottom row. As instance 3's closest common ancestor with the query, "Cars", is farther in the hierarchical tree Fig. 3. This is why $\mathcal{H}\text{-rank}^+(2)$ is greater on the top row $(5/3)$ than on the bottom row $(4/3)$.

fine-grained label as the query but still share some coarser semantics, which is not the case for rank^+.

From $\mathcal{H}\text{-rank}^+$ in Eq. (18) we define the Hierarchical Average Precision, $\mathcal{H}$-AP:

$$\mathcal{H}\text{-AP} = \frac{1}{\sum_{k\in\Omega^+}\text{rel}(k)} \sum_{k\in\Omega^+} \frac{\mathcal{H}\text{-rank}^+(k)}{\text{rank}(k)} \tag{19}$$

Eq. (19) extends the AP to non-binary labels. We replace rank^+ by our hierarchical rank $\mathcal{H}\text{-rank}^+$ and the term $|\Omega^+|$ is replaced by $\sum_{k\in\Omega^+}\text{rel}(k)$ for proper normalization (both representing the "sum of positives", see more details in Sec. B-B1).

$\mathcal{H}$-AP extends the desirable properties of the AP. It evaluates the quality of a ranking by: i) penalizing inversions of instances that are not ranked in decreasing order of relevances with respect to the query, ii) giving stronger emphasis to inversions that occur at the top of the ranking. Finally, we can observe that, by this definition, $\mathcal{H}$-AP is equal to the AP in the binary setting $(L=1)$. This makes $\mathcal{H}$-AP a *consistent generalization* of AP (details in Sec. B-B2).

1) Relevance function design: The relevance $\text{rel}(k)$ defines how "similar" an instance $k\in\Omega^{(l)}$ is to the query q. While $\text{rel}(k)$ might be given as input in information retrieval datasets [66], [67], we need to define it based on the hierarchical tree in our case. We want to enforce the constraint that the relevance decreases when going up the tree, *i.e.* $\text{rel}(k) > \text{rel}(k')$ for $k\in\Omega^{(l)}$, $k'\in\Omega^{(l')}$ and $l > l'$. To do so, we assign a total weight of $(l/L)^\alpha$ to each semantic level l, where $\alpha\in\mathbb{R}^+$ controls the decrease rate of similarity in the tree. For example for $L=3$ and $\alpha=1$, the total weights for each level are 1, $\frac{2}{3}$, $\frac{1}{3}$ and 0. The instance relevance $\text{rel}(k)$ is normalized by the cardinal of $\Omega^{(l)}$:

$$\text{rel}(k) = \frac{(l/L)^\alpha}{|\Omega^{(l)}|} \text{ if } k\in\Omega^{(l)} \tag{20}$$

We set $\alpha=1$ in Eq. (20) for the $\mathcal{H}$-AP metric and in our main experiments. Setting α to larger values supports better performances on fine-grained levels as their relevances will relatively increase. This variant is discussed in Sec. VII-C. Other definitions of the relevance are possible, *e.g.* an interesting option for the relevance enables to recover a weighted sum of AP, denoted as $\sum w\text{AP} := \sum_{l=1}^{L} w_l \cdot \text{AP}^{(l)}$ (supplementary Sec. B-B3), *i.e.* the weighted sum of AP is a particular case of $\mathcal{H}$-AP.

2) Hierarchical Average Precision Training for Pertinent Image Retrieval: We define our surrogate loss to optimize $\mathcal{H}$-AP:

$$\mathcal{L}_{\text{Sup-}\mathcal{H}\text{-AP}} = 1 - \frac{1}{M}\sum_{i=1}^{M} \frac{1}{\sum_{k\in\Omega_i^+}\text{rel}(k)} \sum_{k\in\Omega_i^+} \frac{\mathcal{H}\text{-rank}^+(k)}{\text{rank}^+(k)+\text{rank}_s^-(k)} \tag{21}$$

Note that in the hierarchical case $\text{rank}^-(k)$ is the number of instances of relevances $<\text{rel}(k)$ meaning that it may contain images that are similar to some extent to the query. Finally our ranking loss, **H**ierarchical **A**verage **P**recision training for **P**ertinent **I**mag**E** **R**etrieval (HAPPIER), is obtained by adding $\mathcal{L}_{\text{DG}}^*$:

$$\mathcal{L}_{\text{HAPPIER}} = (1-\lambda)\cdot\mathcal{L}_{\text{Sup-}\mathcal{H}\text{-AP}} + \lambda\cdot\mathcal{L}_{\text{DG}}^* \tag{22}$$

B. Application to the NDCG

The NDCG [46], [47] is a common metric in the information retrieval community. The NDCG is defined using a relevance that is not required to be binary:

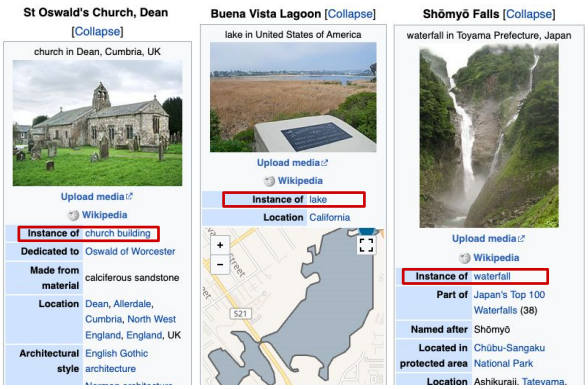
$$\begin{aligned}\text{DCG}_i &= \sum_{k\in\Omega_i^+} \frac{\text{rel}(k)}{\log_2(1+\text{rank}(k))} \\ \text{iDCG}_i &= \max_{\text{rank}} \text{DCG}_i \\ \text{NDCG} &= \frac{1}{M}\sum_{i=1}^{M} \frac{\text{DCG}_i}{\text{iDCG}_i}\end{aligned} \tag{23}$$

We choose the following relevance function for the NDCG: $\text{rel}(k) = 2^l - 1$, if $k\in\Omega^{(l)}$. Using the exponentiation is a standard procedure in information retrieval [47] as it allows to put more emphasis on instances of higher relevance. We then use similarly to other rank losses our SupRank surrogate. We use it to approximate the DCG, and thus the NDCG:

$$\begin{aligned}\text{DCG}_{i,s} &= \sum_{k\in\Omega_i^+} \frac{\text{rel}(k)}{\log_2(1+\text{rank}^+(k)+\text{rank}_s(k))} \\ \mathcal{L}_{\text{Sup-NDCG}} &= 1 - \frac{1}{M}\sum_{i=1}^{M} \frac{\text{DCG}_{i,s}}{\text{iDCG}_i}\end{aligned} \tag{24}$$

Note that once again our surrogate loss, $\mathcal{L}_{\text{Sup-NDCG}}$, is an upper bound on the true loss $1-\text{NDCG}$. Finally our training loss is:

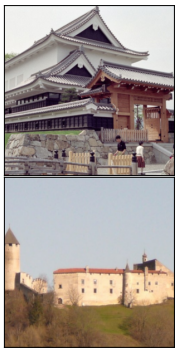
$$\mathcal{L}_{\text{ROD-NDCG}} = (1-\lambda)\cdot\mathcal{L}_{\text{Sup-NDCG}} + \lambda\cdot\mathcal{L}_{\text{DG}}^* \tag{25}$$



(a) Screen captures of Wikimedia Commons webpages.

(b) Waterfall.

(c) Bridge.



(d) Castle.

Fig. 5: Fig. 5a depicts the "Instance of" (within red rectangles), from which we collect hierarchical landmark labels: *e.g. lake*, *waterfall*, *mosque*. Figs. 5b to 5d illustrate some of the supercategories of our $\mathcal{H}$-GLDv2 dataset.

VI. Hierarchical Landmark Dataset

One of the most popular domains for image retrieval research is that of human-made and natural landmarks [36], [68]–[71]. In this work, we introduce for the first time a hierarchical dataset in this domain: $\mathcal{H}$-GLDv2, building on top of the Google Landmarks Dataset v2 (GLDv2) [36], which is the largest and most diverse landmark dataset. In the following, we present our process to semi-automatically annotate GLDv2 with an initial scraping of hierarchical labels from Wikimedia Commons, and a 2-step post-processing of the supercategories. We illustrate some of the created groups in Figs. 5b to 5d. These hierarchical labels are released under the CC BY 4.0 license.

A. Scraping Wikimedia Commons

The landmarks from GLDv2 are sourced from Wikimedia Commons, the world's largest crowdsourced collection of landmark photos. After careful inspection, we find that many of the landmarks in GLDv2 can be associated to supercategories by leveraging the "Instance of" annotations available in Wikimedia Commons – see Fig. 5a. Out of the original $203k$ landmarks in GLDv2-train, we were able to scrape supercategories for $129.1k$. For the $101k$ landmarks in GLDv2-index, we were able to scrape supercategories for $68.1k$. A lightweight manual cleaning process was then applied to remove landmarks assigned to more than one supercategory and those with irrelevant supercategories (*e.g.*, supercategories named "Wikimedia category" or "Wikimedia disambiguation page"). Approximately 0.25% of landmarks end up being removed in this process, leading to a total number of selected landmarks of $128.8k$ and $67.9k$ for the train and index dataset splits, respectively. The number of unique scraped supercategories is $5.7k$.

B. Post-processing supercategories

The scraped supercategories are noisy and do not have the same level of granularity, *e.g.* "church building" *v.s.* "church building (1172–1954)". To mitigate this issue after the scraping we perform a two step post-processing to obtain the final supercategories.

1) **K-means clustering:** We first encode all the labels using the CLIP [72] textual encoder. We perform a k-means on the latent representations. This initial clustering allows to show different prominent categories, *e.g.* "Church", "Castle" *etc.*
2) **Manual verification:** We manually assess the obtained clusters based on the scraped label names. We create semantic groups by dividing the k-means clusters into sub-clusters. This leads to 78 supercategories that we further group into human-made and natural landmarks. Two expert annotators comprehensively reviewed the final clusters manually and filtered them to produce a high-quality dataset.

C. Discussion and limitations

$\mathcal{H}$-GLDv2 is a large scale dataset we were thus not able to manually check all images. This leads to a dataset that can have some noise. We release along with $\mathcal{H}$-GLDv2 the scraped labels to allow further work on the "supercategories". Another difficulty of $\mathcal{H}$-GLDv2 is the ambiguity of some supercategories. For instance, the bottom image of Fig. 5c is labeled as "Bridge", however it could be labeled as "River", another supercategory. Finally, there is an imbalance between supercategories that comes from the classes represented in GLDv2 [36]. We report first results in Sec. VII-C3 of models trained on our $\mathcal{H}$-GLDv2 dataset.

VII. Experiments

A. Standard image retrieval.

In this section, we compare our methods on the standard image retrieval setup, *i.e.* $\mathrm{rel}(x_i,x_j)\in\{0,1\}$, and report fine-grained metrics. We use publicly available implementations of all baselines and run all experiments under the same settings. We use a ResNet-50 backbone with average pooling, a normalization layer without affine parameters and a projection head that reduces the dimension from 2048 to 512. We use a batch size of 256 by sampling 4 images per class and the hierarchical sampling of [24] for SOP, with resolution 224×224, standard data augmentation (random resize crop, horizontal flipping), the Adam optimizer (with learning rate of $5\cdot 10^{-5}$ on SOP and $1\cdot 10^{-5}$ on iNaturalist, with cosine decay) and train for 100 epochs.

1) Comparison to AP approximations: In Tab. I, we compare ROADMAP to AP loss approximations including soft-binning approaches Fast-AP [24] and SoftBin-AP [25], the generic solver BlackBox-AP [32], and the smooth rank approximation [27]. We observe that ROADMAP outperforms all the current AP approximations by a large margin. The gains are especially pronounced on the large-scale dataset iNaturalist.

TABLE I: Comparison between ROADMAP and state-of-the-art AP ranking based methods.

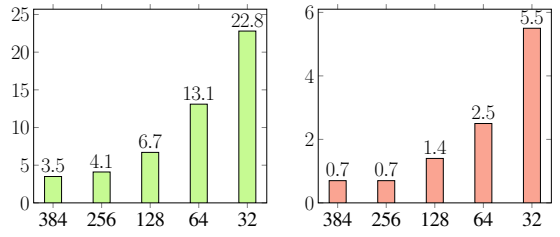
	SOP		iNaturalist	
Method	R@1	mAP@R	R@1	mAP@R
Fast-AP [24]	77.8	50.5	59.9	24.0
SoftBin-AP [25]	79.7	52.7	63.6	25.4
BlackBox-AP [32]	80.0	53.1	52.3	15.2
Smooth-AP [27]	80.9	54.3	67.3	26.5
ROADMAP	**81.9**	**55.7**	**71.8**	**29.5**

2) Ablation study.: To investigate more in-depth the impact of the two components of our framework, we perform ablation studies in Tab. II. We show the improvements against Smooth-AP [27] and Smooth-R@k [28] when replacing the sigmoid by SupRank Eq. (10), and the use of $\mathcal{L}_{\text{DG}}$ Eq. (7) or $\mathcal{L}^*_{\text{DG}}$ Eq. (8). We can see that both $\mathcal{L}_{\text{Sup-AP}}$ and $\mathcal{L}_{\text{Sup-R@k}}$ consistently improve performances over the baselines, +0.5pt mAP@R on SOP and +1pt mAP@R on iNaturalist for both Sup-AP and Sup-R@k. Both $\mathcal{L}_{\text{DG}}$ and $\mathcal{L}^*_{\text{DG}}$ improve over the smooth surrogates, with strong gains on iNaturalist, *e.g.* $\mathcal{L}^*_{\text{DG}}$ improves by +2.9pt R@1 over Sup-AP and +3.7pt R@1 over Sup-R@k. This is because the batch vs. dataset size ratio $\frac{B}{N}$ is tiny ($\sim 8\cdot10^{-4} \ll 1$), making the decomposability gap in Eq. (6) huge. On SOP $\mathcal{L}_{\text{DG}}$ and $\mathcal{L}^*_{\text{DG}}$ work similarly, however on iNat $\mathcal{L}^*_{\text{DG}}$ performs far better than $\mathcal{L}_{\text{DG}}$. In the following, we choose to keep only $\mathcal{L}^*_{\text{DG}}$.

TABLE II: Ablation study of the two components of our framework.

			SOP		iNaturalist	
Method	rank	*DG*	R@1	mAP@R	R@1	mAP@R
Smooth-AP	sigmoid	✗	80.9	54.3	67.3	26.5
Sup-AP	SupRank	✗	81.2	54.8	68.9	27.5
ROADMAP	SupRank	$\mathcal{L}_{\text{DG}}$	81.7	**55.7**	69.1	27.6
		$\mathcal{L}^*_{\text{DG}}$	**81.9**	**55.7**	**71.8**	**29.5**
Smooth-R@k	sigmoid	✗	80.5	53.7	66.4	25.5
Sup-R@k	SupRank	✗	80.7	54.2	68.2	26.4
ROD-R@k	SupRank	$\mathcal{L}_{\text{DG}}$	**82.4**	**56.6**	69.3	27.0
		$\mathcal{L}^*_{\text{DG}}$	81.9	55.8	**71.9**	**29.8**

3) Analysis on decomposability: The decomposability gap depends on the batch size Eq. (6). To illustrate this, we monitor on Fig. 6 the relative improvement when adding $\mathcal{L}^*_{\text{DG}}$ to $\mathcal{L}_{\text{Sup-AP}}$ as the batch size decreases. We can see that the relative improvement becomes larger as the batch size gets smaller. This confirms our intuition that the decomposability loss $\mathcal{L}^*_{\text{DG}}$ has a stronger effect on smaller batch sizes, for which the AP estimation is noisier and DG 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) iNaturalist (b) SOP

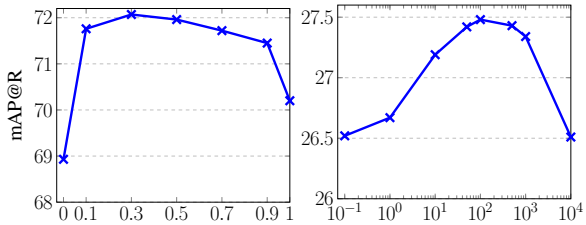
Fig. 6: Relative increase of mAP@R *v.s.* batch size when adding $\mathcal{L}^*_{\text{DG}}$ to $\mathcal{L}_{\text{Sup-AP}}$.

In Tab. III we compare ROADMAP to the cross-batch memory [31] (XBM) which is used to reduce the gap between batch-AP and global AP. We use XBM with a batch size of 128 and store all the dataset, and use the setup described previously otherwise. ROADMAP outperforms XBM both on SOP and iNaturalist with gains more pronounced on iNaturalist with +12.5pt R@1 and +11 mAP@R. $\mathcal{L}^*_{\text{DG}}$ allows us to train models even with smaller batches.

TABLE III: Comparison between XBM [31] and ROADMAP.

	SOP		iNaturalist	
Method	R@1	mAP@R	R@1	mAP@R
XBM [31]	80.6	54.9	59.3	18.5
ROADMAP	**81.9**	**55.7**	**71.8**	**29.5**

4) ROADMAP hyper-parameters: We demonstrate the robustness of our framework to hyper-parameters in Fig. 7. Firstly, Fig. 7a illustrates the complementarity between the two terms of $\mathcal{L}_{\text{ROADMAP}}$. For $0<\lambda<1$, $\mathcal{L}_{\text{ROADMAP}}$ outperforms both $\mathcal{L}_{\text{Sup-AP}}$ and $\mathcal{L}^*_{\text{DG}}$. While we use $\lambda=0.1$ in our experiments, hyper-parameter tuning could yield better results, *e.g.* with $\lambda=0.3$ $\mathcal{L}_{\text{ROADMAP}}$ has 72.1 R@1 *v.s.* 71.8 R@1 reported in Tab. I. Secondly, Fig. 7b shows the influence of the slope ρ that controls the linear regime in H^-. As shown in Fig. 7b, the improvement is important and stable in $[10,100]$. Note that $\rho>1$ already improves the results compared to $\rho=0$ in [27]. There is a decrease when $\rho \gg 10^3$ probably due to the high gradient that takes over the signal for correctly ranked samples.



(a) R@1 *v.s.* λ for $\mathcal{L}_{\text{ROADMAP}}$ (b) mAP@R *v.s.* ρ for $\mathcal{L}_{\text{Sup-AP}}$

Fig. 7: Robustness to hyper-parameters on iNaturalist.

B. Comparison to state-of-the-art

In this section, we compare our AP approximation method, ROADMAP, to state-of-the-art methods, on SOP, CUB, and iNaturalist. We use ROADMAP with a memory [31] to virtually increase the batch size. Note that using batch memory is less computationally expensive than methods such as [28] which trade computational time for memory footprint by using two forward passes. We apply ROADMAP on both a convolutional backbone, ResNet-50 with GeM pooling [68] and layer normalization, and Vision transformer models [77], DeiT-S [78] (Imagenet-1k pre-trained as in [76]) and ViT-B (Imagenet21k pre-trained as in [28]). For convolutional backbones, we choose to keep the standard images of size 224×224 for both training and inference on SOP and iNaturalist, and use more recent settings [6], [15] for CUB and use images of size 256×256. Vision transformers experiments use images of size 224×224.

In Tab. IV, using convolutional backbones, ROADMAP outperforms most state-of-the-art methods when evaluated at different (standard) R@k. As ROADMAP optimizes directly the evaluation metrics, it outperforms metric learning and classification-based methods, *e.g.* +1.4pt R@1 on SOP compared to Triplet SCT [6] or +1.9pt R@1 on SOP *v.s.* ProxyNCA++ [15]. ROADMAP also outperforms R@k [28] with +1.2pt R@1 on SOP and +1.3pt R@1 on iNaturalist. This is impressive as R@k [28] uses a strong setup, *i.e.* a batch size of 4096 and Similarity mixup. On the

TABLE IV: Comparison of state-of-the-art performances on R@K from the literature on SOP, CUB, and iNaturalist with the proposed ROADMAP. Except for the ViT categories, 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 [5]	512	72.7	86.2	93.8	63.6	74.4	83.1	90.0	58.1	75.5	86.8	90.7
	MS [9]	512	78.2	90.5	96.0	65.7	77.0	86.3	91.2	-	-	-	-
	SEC [73]	512	78.7	90.8	96.6	68.8	79.4	87.2	92.5	-	-	-	-
	HORDE [74]	512	80.1	91.3	96.2	66.8	77.4	85.1	91.0	-	-	-	-
	XBM [31]	128	80.6	91.6	96.2	65.8	75.9	84.0	89.9	-	-	-	-
	Triplet SCT [6]	512/64	81.9	92.6	96.8	57.7	69.8	79.6	87.0	-	-	-	-
Classification	ProxyNCA [10]	512	73.7	-	-	49.2	61.9	67.9	72.4	61.6	77.4	87.0	90.6
	ProxyGML [14]	512	78.0	90.6	96.2	66.6	77.6	86.4	-	-	-	-	-
	NSoftmax [11]	512	78.2	90.6	96.2	61.3	73.9	83.5	90.0	-	-	-	-
	NSoftmax [11]	2048	79.5	91.5	96.7	65.3	76.7	85.4	91.8	-	-	-	-
	Cross-Entropy [75]	2048	81.1	91.7	96.3	69.2	79.2	86.9	91.6	-	-	-	-
	ProxyNCA++ [15]	512	80.7	92.0	96.7	69.0	79.8	87.3	92.7	-	-	-	-
	ProxyNCA++ [15]	2048	81.4	92.4	96.9	**72.2**	**82.0**	**89.2**	**93.5**	-	-	-	-
Ranking	FastAP [24]	512	76.4	89.0	95.1	-	-	-	-	60.6	77.0	87.2	90.6
	BlackBox [32]	512	78.6	90.5	96.0	64.0	75.3	84.1	90.6	62.9	79.4	88.7	91.7
	SmoothAP [27]	512	80.1	91.5	96.6	-	-	-	-	67.2	81.8	90.3	93.1
	R@k [28]	512	82.8	92.9	97.0	-	-	-	-	71.2	84.0	91.3	93.6
	R@k + SiMix [28]	512	82.1	92.8	97.0	-	-	-	-	71.8	84.7	91.9	94.3
	ROADMAP (ours)	512	**83.3**	**93.6**	**97.4**	69.4	79 4	87.2	92.1	**73.1**	**85.7**	**92.7**	**94.8**
DeiT-S	IRT_R [76]	384	84.2	93.7	97.3	76.6	85.0	91.1	94.3	-	-	-	-
	ROADMAP (ours)	384	**85.2**	**94.5**	**97.9**	**77.6**	**86.2**	**91.6**	**95.0**	**74.7**	**86.9**	**93.4**	**95.4**
ViT-B	R@k + SiMix [28]	512	88.0	96.1	98.6	-	-	-	-	83.9	92.1	95.9	97.2
	ROADMAP (ours)	512	**88.4**	**96.4**	**98.7**	**86.8**	**91.7**	**94.6**	**96.5**	**85.1**	**93.0**	**96.6**	**97.7**

small-scale dataset CUB, our method is competitive with methods such as ProxyNCA++ with the same embedding size of 512.

Finally, we show that ROADMAP also improves Vision Transformers for image retrieval. With DeiT-S, ROADMAP outperforms [76] on both SOP and CUB by +1pt R@1, this again shows the interest of directly optimizing the metrics rather than the pair loss of [31] used in [76]. With ViT-B, ROADMAP outperforms [28] by +0.4pt R@1 and +1.2pt R@1 on SOP and iNaturalist respectively. We attribute this to the fact that our loss is an actual upper bound of the metric, in addition to our decomposability loss.

C. Hierarchical Results

In this section, we show results on the hierarchical settings and use the labels as described in the additional context of Sec. VI. We report results using the experimental setting of Sec. VII-A. Additionally to hierarchical metrics NDCG and $\mathcal{H}$-AP, we report ASI which is defined in Sec. C-A1.

On Tab. V, we show that HAPPIER significantly outperforms methods trained on the fine-grained level only, with a gain on $\mathcal{H}$-AP over the best performing methods of +16.4pt $\mathcal{H}$-AP on SOP, +13pt on iNat-base and 10.7pt on iNat-full. HAPPIER also exhibits significant gains compared to hierarchical methods. On $\mathcal{H}$-AP, HAPPIER has important gains on all datasets (*e.g.* +6.3pt on SOP, +4.2pt on iNat-base over the best competitor), but also on ASI and NDCG. This shows the strong generalization of the method on standard metrics. Compared to the recent CSL loss [33], we observe a consistent gain over all metrics and datasets, *e.g.* +6pt on $\mathcal{H}$-AP, +8pt on ASI and +2.6pts on NDCG on SOP. This shows the benefits of optimizing a well-behaved hierarchical metric compared to an ad-hoc proxy method. We observe similar trends on the three large scale hierarchical benchmarks DyML [33] in Tab. VIII, that include a Vehicle re-ID dataset, a wild-life dataset and a retail dataset.

Furthermore, we can see that HAPPIER performs on-par to the best methods for standard image retrieval when considering fine-grained metrics. HAPPIER has 81.0 R@1 on SOP *v.s.* 81.4 R@1 for NCA++, and even performs slightly better on iNat-base with 70.7 R@1 *v.s.* 70.2 R@1 for NSM. Finally, our variant HAPPIER_F for $\alpha > 1$ Sec. V-A1, performs as expected (α is 5 on SOP and 3 on iNat-base/full): it is a strong method for fine-grained image retrieval, and still outperforms standard methods on hierarchical metrics.

1) Detailed evaluation: HAPPIER performs well on the overall hierarchical metrics because it performs well on *all* the hierarchical level. We illustrate this on Tab. VI which reports the different methods' performances on all semantic hierarchy levels on iNat-full. We evaluate HAPPIER and HAPPIER_F. HAPPIER optimizes the overall hierarchical performance, while HAPPIER_F is meant to be optimal at the fine-grained level without sacrificing coarser levels. The satisfactory behavior and the two optimal regimes of HAPPIER and HAPPIER_F are confirmed on iNat-full: HAPPIER gives the best results on coarser levels (from "Class"), while being very close to the best results on finer ones. HAPPIER_F gives the best results at the finest levels, even outperforming very competitive fine-grained baselines. HAPPIER also outperforms CSL [33] on all semantic levels, *e.g.* +5pt on the fine-grained AP ("Species") and +3pt on the coarsest AP ("Kingdom"). We show the detailed evaluation on SOP and iNat-base in Sec. C-A3.

2) Model analysis: We showcase the different behavior and the robustness of HAPPIER when changing the hyper-parameters. Fig. 8a studies the impact of α for setting the relevance in Eq. (20). α controls the balance between the relevance weight allocated to each level. Increasing α puts more emphasis on the fine-grained levels, on the contrary, diminishing its value will put an equal contribution to all levels. This is illustrated in Fig. 8a: increasing α improves the AP at the fine-grained level on iNat-base. Fig. 8a shows that one can use α to obtain a range of performances for desired applications.

We measure the impact in Fig. 8b of λ for weighting $\mathcal{L}^s_{\mathcal{H}\text{-AP}}$ and $\mathcal{L}_{\text{DG}}$ in HAPPIER: we observe a stable increase in $\mathcal{H}$-AP

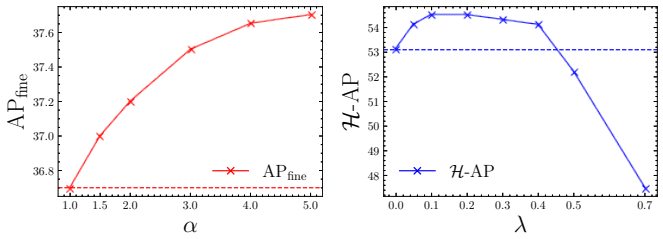
TABLE V: Comparison of HAPPIER on SOP and iNat-base/full. Best results in **bold**, second best underlined.

	Method	SOP					iNat-base					iNat-full				
		R@1	AP	$\mathcal{H}$-AP	ASI	NDCG	R@1	AP	$\mathcal{H}$-AP	ASI	NDCG	R@1	AP	$\mathcal{H}$-AP	ASI	NDCG
Fine	Triplet SH [5]	79.8	59.6	42.2	22.4	78.8	66.3	33.3	39.5	63.7	91.5	66.3	33.3	36.1	59.2	89.8
Fine	NSM [11]	81.3	61.3	42.8	21.1	78.3	70.2	37.6	38.0	51.6	88.9	70.2	**37.6**	33.3	51.7	88.2
Fine	NCA++ [15]	81.4	61.7	43.0	21.5	78.4	67.3	35.2	39.5	57.0	90.1	67.3	35.2	35.3	55.7	89.0
Fine	Smooth-AP [27]	80.9	60.8	42.9	20.6	78.2	67.3	35.2	41.3	64.2	91.9	67.3	35.2	37.2	60.1	90.1
Hier.	ΣTL$_{\text{SH}}$ [5]	78.3	57.6	53.1	53.3	89.2	54.7	21.3	44.0	87.4	96.4	52.9	19.7	39.9	85.5	92.0
Hier.	ΣNSM [11]	79.4	58.4	50.4	49.7	87.0	69.5	37.5	47.9	75.8	94.4	67.2	36.1	46.9	74.2	**93.8**
Hier.	ΣNCA++ [15]	76.3	54.5	49.5	52.8	87.8	64.2	35.4	48.9	78.7	95.0	67.4	36.3	44.7	74.3	92.6
Hier.	CSL [33]	79.4	58.0	52.8	57.9	88.1	62.9	30.2	50.1	**89.3**	96.7	59.9	30.4	45.1	84.9	93.0
	ROD-NDCG (ours)	80.5	59.6	58.3	65.0	91.1	70.7	35.9	53.1	87.8	96.6	71.2	36.7	44.8	81.1	93.1
	HAPPIER (ours)	81.0	60.4	**59.4**	**65.9**	**91.5**	70.7	36.7	**54.3**	**89.3**	**96.9**	70.2	36.0	**47.9**	**87.2**	**93.8**
	HAPPIER$_{\text{F}}$ (ours)	**81.8**	**62.2**	52.0	45.9	86.5	**71.6**	**37.8**	43.2	87.0	96.6	**71.4**	**37.6**	40.1	80.0	93.5

TABLE VI: Comparison of HAPPIER *v.s.* fine-grained methods and CSL on iNat-full. Metrics are reported for all 7 semantic levels.

	Method	Species		Genus	Family	Order	Class	Phylum	Kingdom
		R@1	AP	AP	AP	AP	AP	AP	AP
Fine	TL$_{\text{SH}}$ [5]	66.3	33.3	34.2	32.3	35.4	48.5	54.6	68.4
Fine	NSM [11]	70.2	**37.6**	38.0	31.4	28.6	36.6	43.9	63.0
Fine	NCA++ [15]	67.3	37.0	37.9	33.0	32.3	41.9	48.4	66.1
Fine	Smooth-AP [27]	67.3	35.2	36.3	33.5	35.0	49.3	55.8	69.9
Hier.	CSL [33]	59.9	30.4	32.4	36.2	50.7	81.0	87.4	91.3
	HAPPIER (ours)	70.2	36.0	37.0	38.0	**51.9**	**81.3**	**89.1**	**94.4**
	HAPPIER$_{\text{F}}$ (ours)	**70.8**	**37.6**	**38.2**	**38.8**	50.9	76.1	82.2	83.1

with $0 < \lambda < 0.5$ compared to optimizing only $\mathcal{L}^s_{\mathcal{H}\text{-AP}}$, while a drop in performance is observed for $\lambda > 0.5$. This shows the complementarity of $\mathcal{L}^s_{\mathcal{H}\text{-AP}}$ and $\mathcal{L}^*_{\text{DG}}$, and how, when combined, HAPPIER reaches its best performance.



(a) AP$_{\text{fine}}$ vs α in Eq. (20). (b) $\mathcal{H}$-AP *v.s.* λ for $\mathcal{L}_{\text{HAPPIER}}$.

Fig. 8: Impact on iNat-base of α in Eq. (20) for setting the relevance of $\mathcal{H}$-AP (a) and of the λ hyper-parameter on HAPPIER results (b).

3) Hierarchical landmark results: In this section, we report first results on our $\mathcal{H}$-GLDv2 dataset. We run all experiments under the same settings: we use a ResNet-101 with GeM pooling and initialize a linear projection with a PCA [25]. We use a batch size of 256 and train for $\sim$55k steps with Adam and a learning rate of 10^{-5} decayed using a cosine schedule. We report the mAP@100 [36], and the hierarchical metrics $\mathcal{H}$-AP, ASI and NDCG.

TABLE VII: Comparison of ROADMAP and HAPPIER *v.s.* baselines on $\mathcal{H}$-GLDv2.

Method	mAP@100	$\mathcal{H}$-AP	ASI	NDCG
SoftBin [25]	39.0	35.2	74.6	94.4
Smooth-AP [27]	42.5	37.3	76.9	94.7
R@k [28]	41.6	36.8	77.1	94.7
ROADMAP	42.9	37.0	75.0	94.4
CSL [33]	37.5	36.2	**85.4**	**95.7**
HAPPIER	41.6	**38.8**	83.8	**95.7**
HAPPIER$_{\text{F}}$	**43.7**	38.3	77.5	94.8

In Tab. VII we report the results of ROADMAP and HAPPIER *v.s.* other fine-grained methods and hierarchical methods. Tab. VII demonstrates once again the interest of our AP surrogate, ROADMAP and HAPPIER$_{\text{F}}$ perform the best on the fine-grained metric mAP@100. Furthermore, HAPPIER has the best hierarchical results. It outperforms ROADMAP by +2.8pt $\mathcal{H}$-AP and +8.8pt ASI. It also outperforms CSL by +2.6pt $\mathcal{H}$-AP.

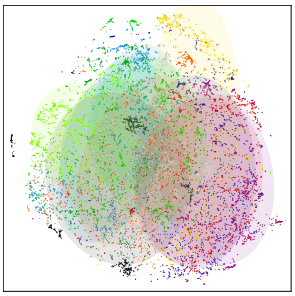
4) Qualitative experiments: We assess qualitatively HAPPIER, including embedding space analysis and visualization of HAPPIER's retrievals.

t-SNE: organization of the embedding space: In Figs. 9a and 9b, we plot using t-SNE [79], [80] how HAPPIER learns an embedding space on SOP ($L = 2$) that is well-organized. We plot the mean vector of each fine-grained class, and we assign the color based on the coarse level. We compare the t-SNE the embedding space of a baseline (Smooth-AP [27]) on Fig. 9a and of HAPPIER in Fig. 9b. We cannot observe any clear clusters for the coarse level on Fig. 9a, whereas we can appreciate the quality of the hierarchical clusters formed on Fig. 9b.

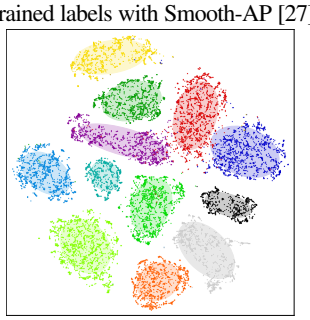
Controlled errors on iNat-base: Finally, we showcase in Figs. 9c and 9d errors of HAPPIER *v.s.* a fine-grained baseline (Smooth-AP) on iNat-base. On Fig. 9c, we illustrate how a model trained with HAPPIER makes less severe mistakes than a model trained only on the fine-grained level. On Fig. 9d, we show an example where both models fail to retrieve the correct fine-grained instances, however the model trained with HAPPIER retrieves images that are semantically more similar to the query. This shows the robustness of HAPPIER's ranking.

VIII. Conclusion

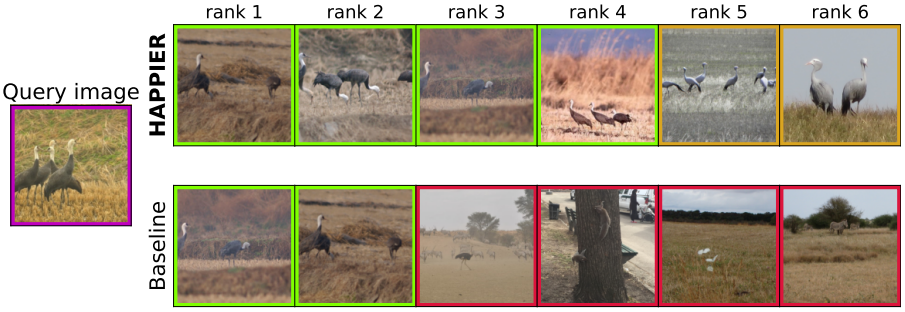
In this work we have introduced a general framework for rank losses optimization. It tackles two issues of rank losses



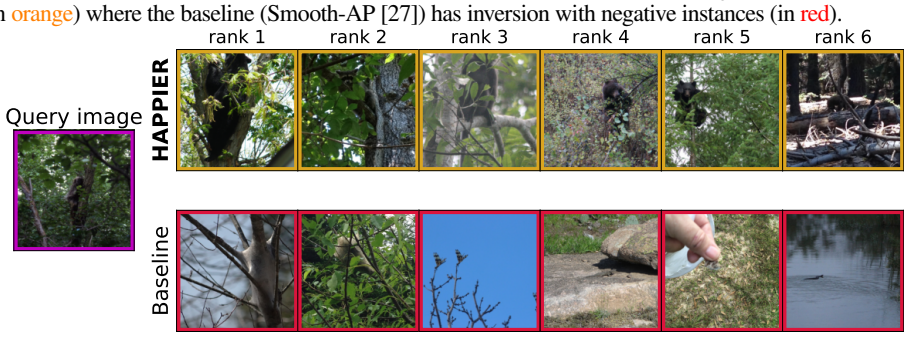
(a) t-SNE visualization of a model trained on fine-grained labels with Smooth-AP [27].



(b) t-SNE visualization of a model trained with **HAPPIER**.



(c) **HAPPIER** makes less severe mistakes. Its inversions are with instances sharing the same coarse label (in orange) where the baseline (Smooth-AP [27]) has inversion with negative instances (in red).



(d) When models fail to retrieve the correct fine-grained images, **HAPPIER** still retrieves images with the same coarse label (in orange) whereas the baseline (Smooth-AP [27]) retrieves negative instances (in red).

optimization: 1) non-differentiability using smooth and upper bound rank approximation, 2) non-decomposability using an additional objective. We apply our framework to both fine-grained, by optimizing the AP and R@k, and hierarchical image retrieval, by optimizing the NDCG and the introduced $\mathcal{H}$-AP. We show that using our framework outperforms other rank loss surrogates on several standard fine-grained and hierarchical image retrieval benchmarks, including the hierarchical landmark dataset we introduce in this work. We also show that our framework sets state-of-the-art results for fine-grained image retrieval.

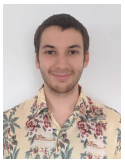
ACKNOWLEDGMENT

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

REFERENCES

[1] E. Xing, M. Jordan, S. J. Russell, and A. Ng, "Distance metric learning with application to clustering with side-information," in *NeurIPS*, 2003.
[2] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *CVPR*, 2006.
[3] F. Radenovic, G. Tolias, and O. Chum, "CNN image retrieval learns from bow: Unsupervised fine-tuning with hard examples," in *ECCV*, 2016.
[4] A. Gordo, J. Almazán, J. Revaud, and D. Larlus, "End-to-end learning of deep visual representations for image retrieval," *Int. J. Comput. Vis.*, 2017.
[5] C.-Y. Wu, R. Manmatha, A. J. Smola, and P. Krahenbuhl, "Sampling matters in deep embedding learning," in *ICCV*, 2017.
[6] H. Xuan, A. Stylianou, X. Liu, and R. Pless, "Hard negative examples are hard, but useful," in *ECCV*, 2020.
[7] K. Sohn, "Improved deep metric learning with multi-class n-pair loss objective," in *NeurIPS*, 2016.
[8] M. T. Law, N. Thome, and M. Cord, "Learning a distance metric from relative comparisons between quadruplets of images," *Int. J. Comput. Vis.*, 2017.
[9] X. Wang, X. Han, W. Huang, D. Dong, and M. R. Scott, "Multi-similarity loss with general pair weighting for deep metric learning," in *CVPR*, 2019.
[10] Y. Movshovitz-Attias, A. Toshev, T. K. Leung, S. Ioffe, and S. Singh, "No fuss distance metric learning using proxies," in *ICCV*, 2017.
[11] A. Zhai and H. Wu, "Classification is a strong baseline for deep metric learning," *BMVC*, 2018.
[12] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu, "Cosface: Large margin cosine loss for deep face recognition," in *CVPR*, 2018.
[13] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *CVPR*, 2019.
[14] Y. Zhu, M. Yang, C. Deng, and W. Liu, "Fewer is more: A deep graph metric learning perspective using fewer proxies," in *NeurIPS*, 2020.
[15] E. W. Teh, T. DeVries, and G. W. Taylor, "Proxynca++: Revisiting and revitalizing proxy neighborhood component analysis," in *ECCV*, 2020.
[16] Y. Yue, T. Finley, F. Radlinski, and T. Joachims, "A support vector method for optimizing average precision," in *SIGIR*, 2007.
[17] B. Mcfee and G. Lanckriet, "Metric learning to rank," in *ICML*, 2010.
[18] P. Mohapatra, M. Rolínek, C. Jawahar, V. Kolmogorov, and M. P. Kumar, "Efficient optimization for rank-based loss functions," in *CVPR*, 2018.
[19] T. Durand, N. Thome, and M. Cord, "Exploiting negative evidence for deep latent structured models," *TPAMI*, 2019.
[20] M. Vlastelica, A. Paulus, V. Musil, G. Martius, and M. Rolínek, "Differentiation of blackbox combinatorial solvers," in *ICLR*, 2020.
[21] E. Ustinova and V. Lempitsky, "Learning deep embeddings with histogram loss," in *NeurIPS*, 2016.
[22] K. He, F. Cakir, S. A. Bargal, and S. Sclaroff, "Hashing as tie-aware learning to rank," in *CVPR*, 2018.
[23] K. He, Y. Lu, and S. Sclaroff, "Local descriptors optimized for average precision," in *CVPR*, 2018.
[24] F. Cakir, K. He, X. Xia, B. Kulis, and S. Sclaroff, "Deep metric learning to rank," in *CVPR*, 2019.
[25] J. Revaud, J. Almazán, R. S. Rezende, and C. R. d. Souza, "Learning with average precision: Training image retrieval with a listwise loss," in *ICCV*, 2019.
[26] M. Engilberge, L. Chevallier, P. Perez, and M. Cord, "Sodeep: A sorting deep net to learn ranking loss surrogates," in *CVPR*, 2019.
[27] A. Brown, W. Xie, V. Kalogeiton, and A. Zisserman, "Smooth-ap: Smoothing the path towards large-scale image retrieval," in *ECCV*, 2020.
[28] Y. Patel, G. Tolias, and J. Matas, "Recall@ k surrogate loss with large batches and similarity mixup," in *CVPR*, 2022.
[29] W. Ge, "Deep metric learning with hierarchical triplet loss," in *ECCV*, 2018.
[30] Y. Suh, B. Han, W. Kim, and K. M. Lee, "Stochastic class-based hard example mining for deep metric learning," in *CVPR*, 2019.
[31] X. Wang, H. Zhang, W. Huang, and M. R. Scott, "Cross-batch memory for embedding learning," in *CVPR*, 2020.
[32] M. Rolínek, V. Musil, A. Paulus, M. Vlastelica, C. Michaelis, and G. Martius, "Optimizing rank-based metrics with blackbox differentiation," in *CVPR*, 2020.
[33] Y. Sun, Y. Zhu, Y. Zhang, P. Zheng, X. Qiu, C. Zhang, and Y. Wei, "Dynamic metric learning: Towards a scalable metric space to accommodate multiple semantic scales," in *CVPR*, 2021.
[34] W. Zheng, Y. Huang, B. Zhang, J. Zhou, and J. Lu, "Dynamic metric learning with cross-level concept distillation," in *ECCV*, 2022.

[35] E. Ramzi, N. Audebert, N. Thome, C. Rambour, and X. Bitot, "Hierarchical average precision training for pertinent image retrieval," in *ECCV*, 2022.
[36] T. Weyand, A. Araujo, B. Cao, and J. Sim, "Google landmarks dataset v2-a large-scale benchmark for instance-level recognition and retrieval," in *CVPR*, 2020.
[37] E. Ramzi, N. Thome, C. Rambour, N. Audebert, and X. Bitot, "Robust and decomposable average precision for image retrieval," *NeurIPS*, 2021.
[38] B. Harwood, V. Kumar B G, G. Carneiro, I. Reid, and T. Drummond, "Smart mining for deep metric learning," in *ICCV*, 2017.
[39] F. Faghri, D. J. Fleet, J. R. Kiros, and S. Fidler, "VSE++: improving visual-semantic embeddings with hard negatives," in *BMVC*, 2018.
[40] M. Carvalho, R. Cadène, D. Picard, L. Soulier, N. Thome, and M. Cord, "Cross-modal retrieval in the cooking context: Learning semantic text-image embeddings," in *SIGIR*, 2018.
[41] A. Dhall, A. Makarova, O. Ganea, D. Pavllo, M. Greeff, and A. Krause, "Hierarchical image classification using entailment cone embeddings," in *CVPR Workshops*, 2020.
[42] L. Bertinetto, R. Mueller, K. Tertikas, S. Samangooei, and N. A. Lord, "Making better mistakes: Leveraging class hierarchies with deep networks," in *CVPR*, 2020.
[43] D. Chang, K. Pang, Y. Zheng, Z. Ma, Y.-Z. Song, and J. Guo, "Your" flamingo" is my" bird": Fine-grained, or not," in *CVPR*, 2021.
[44] B. Hjørland, "The foundation of the concept of relevance," *JIST*, 2010.
[45] J. Kekäläinen and K. Järvelin, "Using graded relevance assessments in ir evaluation," *JIST*, 2002.
[46] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of ir techniques," *ACM TOIS*, 2002.
[47] W. B. Croft, D. Metzler, and T. Strohman, *Search engines: Information retrieval in practice*. Addison-Wesley Reading, 2010.
[48] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, "Learning to rank using gradient descent," in *ICML*, 2005.
[49] C. Burges, R. Ragno, and Q. Le, "Learning to rank with nonsmooth cost functions," in *NeurIPS*, 2006.
[50] M. Taylor, J. Guiver, S. Robertson, and T. Minka, "Softrank: Optimizing non-smooth rank metrics," in *WSDM*, 2008.
[51] T. Qin, T.-Y. Liu, and H. Li, "A general approximation framework for direct optimization of information retrieval measures," *Information Retrieval*, 2009.
[52] S. Bruch, M. Zoghi, M. Bendersky, and M. Najork, "Revisiting approximate metric optimization in the age of deep neural networks," in *SIGIR*, 2019.
[53] G. Dupret and B. Piwowarski, "Model based comparison of discounted cumulative gain and average precision," *Journal of Discrete Algorithms*, 2013.
[54] S. E. Robertson, E. Kanoulas, and E. Yilmaz, "Extending average precision to graded relevance judgments," in *SIGIR*, 2010.
[55] G. Dupret and B. Piwowarski, "A user behavior model for average precision and its generalization to graded judgments," in *SIGIR*, 2010.
[56] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The Caltech-UCSD Birds-200-2011 Dataset," Caltech, Tech. Rep., 2011.
[57] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3d object representations for fine-grained categorization," in *3dRR Workshop*, 2013.
[58] Z. Liu, P. Luo, S. Qiu, X. Wang, and X. Tang, "Deepfashion: Powering robust clothes recognition and retrieval with rich annotations," in *CVPR*, 2016.
[59] H. O. Song, Y. Xiang, S. Jegelka, and S. Savarese, "Deep metric learning via lifted structured feature embedding," in *CVPR*, 2016.
[60] G. Van Horn, O. Mac Aodha, Y. Song, Y. Cui, C. Sun, A. Shepard, H. Adam, P. Perona, and S. Belongie, "The inaturalist species classification and detection dataset," in *CVPR*, 2018.
[61] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.
[62] G. A. Miller, "Wordnet: A lexical database for english," *Commun. ACM*, 1995.
[63] E. W. Wilt and A. V. Harrison, "Creating a semantic hierarchy of SUN database object labels using WordNet," in *AI/ML for Multi-Domain Operations*, 2021.
[64] Z. Li, W. Min, J. Song, Y. Zhu, L. Kang, X. Wei, X. Wei, and S. Jiang, "Rethinking the optimization of average precision: Only penalizing negative instances before positive ones is enough," in *AAAI*, 2022.
[65] C. Liao, T. Tsiligkaridis, and B. Kulis, "Supervised metric learning for retrieval via contextual similarity optimization," *arXiv*, 2022.
[66] T. Qin and T. Liu, "Introducing LETOR 4.0 datasets," *CoRR*, 2013.
[67] O. Chapelle and Y. Chang, "Yahoo! learning to rank challenge overview," in *Proceedings of the learning to rank challenge*, 2011.
[68] F. Radenović, A. Iscen, G. Tolias, Y. Avrithis, and O. Chum, "Revisiting oxford and paris: Large-scale image retrieval benchmarking," in *CVPR*, 2018.
[69] D. Chen, G. Baatz, K. Koser, S. Tsai, R. Vedantham, T. Pylvanainen, K. Roimela, X. Chen, J. Bach, M. Pollefeys, B. Girod, and R. Grzeszczuk, "City-Scale Landmark Identification on Mobile Devices," in *CVPR*, 2011.
[70] Y. Avrithis, G. Tolias, and Y. Kalantidis, "Feature Map Hashing: Sub-linear Indexing of Appearance and Global Geometry," in *Proc. ACM MM*, 2010.
[71] A. Torii, R. Arandjelović, J. Sivic, M. Okutomi, and T. Pajdla, "24/7 place recognition by view synthesis," in *Proc. CVPR*, 2015.
[72] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," in *ICML*, 2021.
[73] D. Zhang, Y. Li, and Z. Zhang, "Deep metric learning with spherical embedding," in *NeurIPS*, 2020.
[74] P. Jacob, D. Picard, A. Histace, and E. Klein, "Metric learning with horde: High-order regularizer for deep embeddings," in *ICCV*, 2019.
[75] M. Boudiaf, J. Rony, I. M. Ziko, E. Granger, M. Pedersoli, P. Piantanida, and I. B. Ayed, "A unifying mutual information view of metric learning: cross-entropy vs. pairwise losses," in *ECCV*, 2020.
[76] A. El-Nouby, N. Neverova, I. Laptev, and H. Jégou, "Training vision transformers for image retrieval," *arXiv*, 2021.
[77] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv*, 2020.
[78] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," *arXiv*, 2020.
[79] L. van der Maaten and G. Hinton, "Visualizing high-dimensional data using t-sne," *Journal of Machine Learning Research*, 2008.
[80] D. M. Chan, R. Rao, F. Huang, and J. F. Canny, "Gpu accelerated t-distributed stochastic neighbor embedding," *JPDC*, 2019.
[81] R. Fagin, R. Kumar, and D. Sivakumar, "Comparing top k lists," *JDM*, 2003.



Elias Ramzi is a research scientist at valeo.ai. He obtained a Ph.D in computer vision from Cnam in 2024 and a M.Eng from Supélec in 2020. His current work focus on computer vision and deep learning for autonomous driving.

Nicolas Audebert is an associate professor of computer science at Cnam (Paris, France) since 2019. His research focus on deep representation learning for computer vision and Earth Observation. He graduated with a Ph.D. from the University of South Brittany in 2018 and a M.Eng. from Supélec in 2015.

Clément Rambour is an Assistant Professor at the Cnam (Paris, France) since 2020. He obtained his Ph.D. at Telecom Paris in 2019, specializing in 3D SAR imaging. His research primarily revolves around machine learning and deep learning for image understanding and signal processing, focusing on natural images, remote sensing and healthcare data.

André Araujo is a Senior Staff Research Scientist / Tech Lead Manager at Google DeepMind. His current work focuses on computer vision and machine learning. He graduated with a Ph.D. from Stanford University in 2016. He is a Senior Member of the IEEE.

Xavier Bitot is a project leader at Coexya (France). He graduated with a M.Eng. from École Nationale Superieure de Physique de Strasbourg in 2002. He supervises the Coexya SIP (Paris) entity R&D team on content based text and image classification and retrieval for industrial property.

Nicolas Thome is a full professor at Sorbonne Université (Paris, France). His research interests include machine learning and deep learning for understanding low-level signals, *e.g.* vision, time series, acoustics, *etc.*. His current application domains are targeted towards healthcare, autonomous driving and physics.

APPENDIX A
ADDITIONAL DETAILS ON METHOD

A. Decomposability gap: Average Precision

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

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

We illustrates the decomposability gap, DG 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 Sec. III-C. The motivation behind $\mathcal{L}_{\mathrm{DG}}$ is thus to force the scores of the different batches to be calibrated between mini-batches.

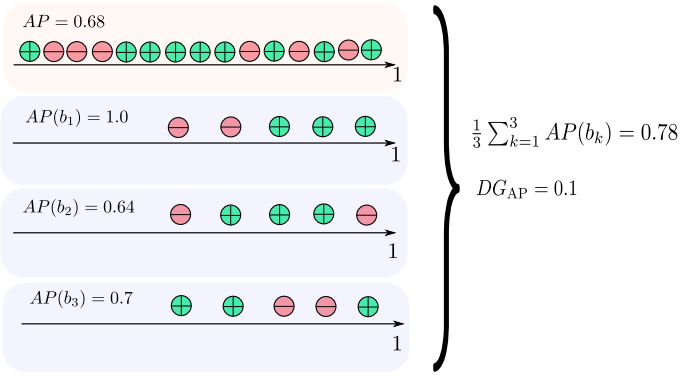


Fig. 10: Illustration of the decomposability gap on a toy dataset.

B. Upper bound on the decomposabilty gap

To formalize this idea, we provide a theoretical analysis of the impact on the global ranking of $\mathcal{L}_{\mathrm{DG}}$ in Eq. (7) for standard image retrieval. Firstly, we can see that if $\mathcal{L}_{\mathrm{DG}} = 0$ on every batch, the overall $\mathcal{M}$ and $\mathcal{M}_b$ is maximal (1), *i.e.* $DG(\boldsymbol{\theta}) = 0$ and we get a decomposable $\mathcal{M}$. In a more general setting, we show that minimizing $\mathcal{L}_{\mathrm{DG}}$ on each batch reduces the decomposability gap, hence improving the decomposability of the $\mathcal{M}$.

Let us consider K batches $\{\mathcal{B}_b\}_{b\in\{1:K\}}$ of batch size B divided in Ω_b^+ positive instances and Ω_b^- negative instances w.r.t. the query $\boldsymbol{q_i}$. To give some insight we assume that $\mathcal{M}_b = 1$. This results in the upper bound of DG given in Eq. (26). This upper bound of the decomposability gap is given in the worst case for the global $\mathcal{M}$: the global ranking is built from the juxtaposition of the batches (see proof bellow).

We can tighten this upper bound by introducing the decomposability loss $\mathcal{L}_{\mathrm{DG}}$ and constraining the scores of positive and negative instances to be well calibrated. On each batch we define the following quantities $E_b^- = \sum_{j\in\Omega_b^-} \mathbb{1}(s_j > \beta)$ which are the number of negative instances that do not respect the constraints and $G_b^- = \sum_{j\in\Omega_b^-} \mathbb{1}(s_j \leq \beta)$ the number of negative instances that do. We similarly define E_b^+ and G_b^+. Giving the upper bound of Eq. (27). $\mathcal{L}_{\mathrm{DG}}$ 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 $\mathcal{M}$.

Proof of Eq. (26)*: Upper bound on the DG with no $\mathcal{L}_{DG}$:* 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. (26) gives an upper bound 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.

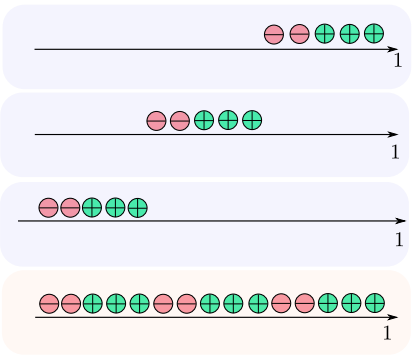


Fig. 11: The worst case when computing the global AP would be that each batch is juxtaposed.

In Eq. (26) 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. (26), 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. (26).

Proof of Eq. (27)*: Upper bound on the DG with $\mathcal{L}_{DG}$:* We refine the upper bound on DG in Eq. (27) by adding $\mathcal{L}_{\mathrm{DG}}$ which calibrates the absolute scores across the mini-batches.

We now write that each positive instance that respects the constraint of $\mathcal{L}_{\mathrm{DG}}$ is ranked after the positive instances of previous batch that respect the constraint giving the following rank$^+$: $j + G_1^+ + \cdots + G_{b-1}^+$, 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 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 rank$^-$: $|\mathcal{N}_i^1| + \cdots + |\mathcal{N}_i^{b-1}|$. Resulting in Eq. (27).

C. 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 $\delta \simeq 0.05$.

without $\mathcal{L}_{\text{DG}}$
$$0 \leq DG \leq 1 - \frac{1}{\sum_{b=1}^{K} |\Omega_b^+|} \left(\sum_{b=1}^{K} \sum_{j=1}^{|\Omega_b^+|} \frac{j + |\Omega_1^+| + \cdots + |\Omega_{b-1}^+|}{j + |\Omega_1^+| + \cdots + |\Omega_{b-1}^+| + |\Omega_1^-| + \cdots + |\Omega_{b-1}^-|} \right) \tag{26}$$

with $\mathcal{L}_{\text{DG}}$
$$0 \leq DG \leq 1 - \frac{1}{\sum_{b=1}^{K} |\Omega_b^+|} \left(\sum_{b=1}^{K} \left[\sum_{j=1}^{G_b^+} \frac{j + G_1^+ + \cdots + G_{b-1}^+}{j + G_1^+ + \cdots + G_{b-1}^+ + E_1^- + \ldots E_{b-1}^-} + \sum_{j=1}^{E_b^+} \frac{j + G_b^+ + |\Omega_1^+| + \cdots + |\Omega_{b-1}^+|}{j + G_b^+ + |\Omega_1^+| + \cdots + |\Omega_{b-1}^+| + |\Omega_1^-| + \cdots + |\Omega_{b-1}^-|} \right] \right) \tag{27}$$

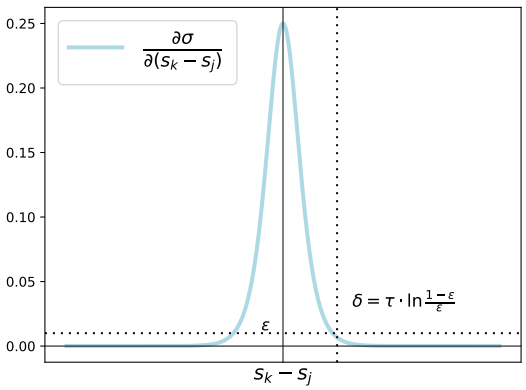


Fig. 12: Gradient of the temperature scaled sigmoid ($\tau = 0.01$) *v.s.* the difference of scores $s_k - s_j$ of a negative pair.

APPENDIX B
ADDITIONAL DETAILS ON $\mathcal{H}$-AP

A. Details on $\mathcal{H}$-rank$^+$

We define the $\mathcal{H}$-rank$^+$ in the main paper as:

$$\mathcal{H}\text{-rank}^+(k) = \text{rel}(k) + \sum_{j \in \Omega^+} \min(\text{rel}(k), \text{rel}(j)) \cdot H(s_j - s_k) \ . \tag{28}$$

We detail in Fig. 13 how the $\mathcal{H}$-rank$^+$ in Eq. (28) is computed in the example from Fig. 3 of the main paper. Given a "Lada #2" query, we set the relevances as follows: if $k \in \Omega^{(3)}$ (*i.e.* k is also a "Lada #2"), $\text{rel}(k) = 1$; if $k \in \Omega^{(2)}$ (*i.e.* k is another model of "Lada"), $\text{rel}(k) = 2/3$; and if $k \in \Omega^{(1)}$ (k is a "Car"), $\text{rel}(k) = 1/3$. Relevance of negatives (other vehicles) is set to 0.

In this instance, $\mathcal{H}\text{-rank}^+(2) = 4/3$ because $\text{rel}(2) = 1$ and $\min(\text{rel}(1), \text{rel}(2)) = \text{rel}(1) = 1/3$. Here, the closest common ancestor in the hierarchical tree shared by the query and instances 1 and 2 is "Cars". For binary labels, we would have $\text{rank}^+(2) = 1$; this would not take into account the semantic similarity between the query and instance 1.

B. Details on $\mathcal{H}$-AP

We define $\mathcal{H}$-AP in the main paper as:

$$\mathcal{H}\text{-AP} = \frac{1}{\sum_{k \in \Omega^+} \text{rel}(k)} \sum_{k \in \Omega^+} \frac{\mathcal{H}\text{-rank}^+(k)}{\text{rank}(k)} \tag{29}$$

We illustrate in Fig. 14 how the $\mathcal{H}$-AP is computed for two rankings. We use the same relevances as in Sec. B-A. The $\mathcal{H}$-AP of the first example is greater (0.78) than of the second one (0.67) because the error is less severe. On the contrary, the AP only considers binary labels and is the same for both rankings (0.45).

One property of AP is that it can be interpreted as the area under the precision-recall curve. $\mathcal{H}$-AP from Eq. (29) can also be interpreted as the area under a hierarchical-precision-recall curve by defining a Hierarchical Recall ($\mathcal{H}$-R@k) and a Hierarchical Precision ($\mathcal{H}$-P@k) as:

$$\mathcal{H}\text{-R@k} = \frac{\sum_{j=1}^{k} \text{rel}(j)}{\sum_{j \in \Omega^+} \text{rel}(j)} \tag{30}$$

$$\mathcal{H}\text{-P@k} = \frac{\sum_{j=1}^{k} \min(\text{rel}(j), \text{rel}(k))}{k \cdot \text{rel}(k)} \tag{31}$$

So that $\mathcal{H}$-AP can be re-written as:

$$\mathcal{H}\text{-AP} = \sum_{k=1}^{|\Omega|} (\mathcal{H}\text{-R@k} - \mathcal{H}\text{-R@k-1}) \times \mathcal{H}\text{-P@k} \tag{32}$$

Eq. (32) recovers Eq. (19) from the main paper, meaning that $\mathcal{H}$-AP generalizes this property of AP beyond binary labels. To further motivate $\mathcal{H}$-AP we will justify the normalization constant for $\mathcal{H}$-AP, and show that $\mathcal{H}$-AP, $\mathcal{H}$-R@k and $\mathcal{H}$-P@k are consistent generalization of AP, R@k, P@k.

1) Normalization constant for $\mathcal{H}$-AP: When all instances are perfectly ranked, all instances j that are ranked before instance k ($s_j \geq s_k$) have a relevance that is higher or equal than k's, *i.e.* $\text{rel}(j) \geq \text{rel}(k)$ and $\min(\text{rel}(j), \text{rel}(k)) = \text{rel}(k)$. So, for each instance k:

$$\begin{aligned}
\mathcal{H}\text{-rank}^+(k) &= \text{rel}(k) + \sum_{j \in \Omega^+} \min(\text{rel}(k), \text{rel}(j)) \cdot H(s_j - s_k) \\
&= \text{rel}(k) + \sum_{j \in \Omega^+} \text{rel}(k) \cdot H(s_j - s_k) \\
&= \text{rel}(k) \cdot \left(1 + \sum_{j \in \Omega^+} H(s_j - s_k) \right) = \text{rel}(k) \cdot \text{rank}(k)
\end{aligned}$$

The total sum $\sum_{k \in \Omega^+} \frac{\mathcal{H}\text{-rank}^+(k)}{\text{rank}(k)} = \sum_{k \in \Omega^+} \text{rel}(k)$. This means that we need to normalize by $\sum_{k \in \Omega^+} \text{rel}(k)$ in order to constrain $\mathcal{H}$-AP between 0 and 1. This results in the definition of $\mathcal{H}$-AP from Eq. (29).

2) $\mathcal{H}$-AP is a consistent generalization of AP: In a binary setting, AP is defined as follows:

$$\text{AP} = \frac{1}{|\Omega^+|} \sum_{k \in \Omega^+} \frac{\text{rank}^+(k)}{\text{rank}(k)} \tag{33}$$

$\mathcal{H}$-AP is equivalent to AP in a binary setting ($L = 1$). Indeed, the relevance function is 1 for fine-grained instances and 0 otherwise in the binary case. Therefore $\mathcal{H}\text{-rank}^+(k) = 1 + \sum_{j \in \Omega^+} H(s_j - s_k)$ which is the same definition as rank$^+$ in AP. Furthermore the normalization constant of $\mathcal{H}$-AP, $\sum_{k \in \Omega^+} \text{rel}(k)$, is equal to the

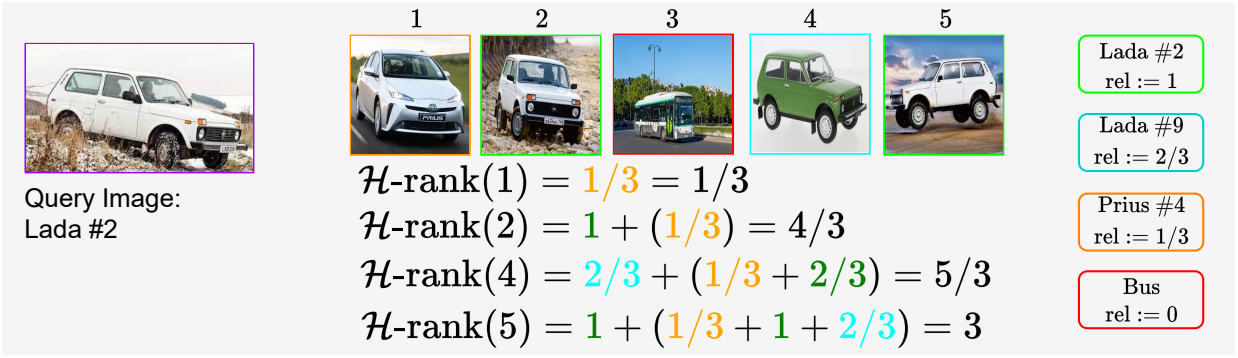


Fig. 13: $\mathcal{H}$-rank$^+$ for each retrieval results given a "Lada #2" query with relevances of Sec. B-A and the hierarchical tree of Fig. 3 in the main paper.

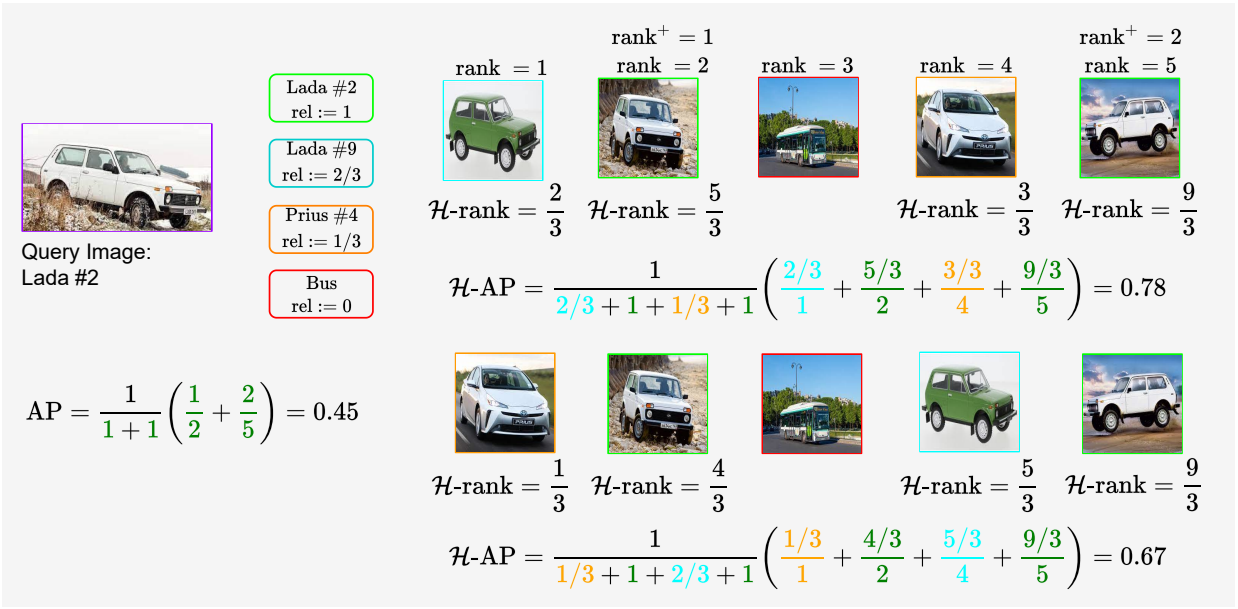


Fig. 14: AP and $\mathcal{H}$-AP for two different rankings when Given a "Lada #2" query and relevances of Sec. B-A. The $\mathcal{H}$-AP of the top row is greater (0.78) than the bottom one's (0.67) as the error in rank$=1$ is less severe for the top row. Whereas the AP is the same for both rankings (0.45).

number of fine-grained instances in the binary setting, *i.e.* $|\Omega^+|$. This means that $\mathcal{H}$-AP$=$AP in this case.

$\mathcal{H}$-R@k is also a consistent generalization of R@k, indeed:

$$\begin{aligned}\mathcal{H}\text{-R@k} &= \frac{\sum_{j=1}^{k}\text{rel}(j)}{\sum_{j\in\Omega^+}\text{rel}(j)} = \frac{\sum_{j=1}^{k}\mathbb{1}(k\in\Omega^+)}{\sum_{j\in\Omega^+}\mathbb{1}(k\in\Omega^+)} \\ &= \frac{\#\text{ number of positive before k}}{|\Omega^+|} \\ &= R@k\end{aligned}$$

Finally, $\mathcal{H}$-P@k is also a consistent generalization of P@k:

$$\begin{aligned}\mathcal{H}\text{-P@k} &= \frac{\sum_{j=1}^{k}\min(\text{rel}(j),\text{rel}(k))}{k\cdot\text{rel}(k)} \\ &= \frac{\#\text{ number of positive before k}}{k} \\ &= P@k\end{aligned}$$

3) Link between $\mathcal{H}$-AP and the weighted average of AP: Let us define the AP for the semantic level $l\geq 1$ as the binary AP with the set of positives being all instances that belong the same level, *i.e.* $\Omega^{+,l}=\bigcup_{q=l}^{L}\Omega^{(q)}$:

$$\text{AP}^{(l)} = \frac{1}{|\Omega^{+,l}|}\sum_{k\in\Omega^{+,l}}\frac{\text{rank}^{+,l}(k)}{\text{rank}(k)}, \text{rank}^{+,l}(k) = 1 + \sum_{j\in\Omega^{+,l}} H(s_j - s_k) \tag{34}$$

Property 1: For any relevance function $\text{rel}(k) = \sum_{p=1}^{l}\frac{w_p}{|\Omega^{+,q}|}$, $k\in\Omega^{(l)}$, with positive weights $\{w_l\}_{l\in[\![1;L]\!]}$ such that $\sum_{l=1}^{L}w_l = 1$:

$$\mathcal{H}\text{-AP} = \sum_{l=1}^{L} w_l \cdot AP^{(l)}$$

i.e. $\mathcal{H}$-AP is equal the weighted average of the AP at all semantic levels.

Proof of Property 1

Denoting $\Sigma w\text{AP} := \sum_{l=1}^{L} w_l \cdot AP^{(l)}$, we obtain from Eq. (34):

$$\Sigma w\text{AP} = \sum_{l=1}^{L} w_l \cdot \frac{1}{|\Omega^{+,l}|} \sum_{k \in \Omega^{+,l}} \frac{\text{rank}^{+,l}(k)}{\text{rank}(k)} \tag{35}$$

We define $\hat{w}_l = \frac{w_l}{|\Omega^{+,l}|}$ to ease notations, so:

$$\Sigma w\text{AP} = \sum_{l=1}^{L} \hat{w}_l \sum_{k \in \Omega^{+,l}} \frac{\text{rank}^{+,l}(k)}{\text{rank}(k)} \tag{36}$$

We define $\mathbb{1}(k,l) = \mathbb{1}\left[k \in \Omega^{+,l}\right]$ so that we can sum over Ω^+ instead of $\Omega^{+,l}$ and inverse the summations. Note that rank does not depend on l, on contrary to $\text{rank}^{+,l}$.

$$\Sigma w\text{AP} = \sum_{l=1}^{L} \sum_{k \in \Omega^+} \frac{\hat{w}_l \cdot \mathbb{1}(k,l) \cdot \text{rank}^{+,l}(k)}{\text{rank}(k)} \tag{37}$$

$$= \sum_{k \in \Omega^+} \sum_{l=1}^{L} \frac{\hat{w}_l \cdot \mathbb{1}(k,l) \cdot \text{rank}^{+,l}(k)}{\text{rank}(k)} \tag{38}$$

$$= \sum_{k \in \Omega^+} \frac{\sum_{l=1}^{L} \mathbb{1}(k,l) \cdot \hat{w}_l \cdot \text{rank}^{+,l}(k)}{\text{rank}(k)} \tag{39}$$

We replace $\text{rank}^{+,l}$ in Eq. (39) with its definition from Eq. (34):

$$\Sigma w\text{AP} = \sum_{k \in \Omega^+} \frac{\sum_{l=1}^{L} \mathbb{1}(k,l) \cdot \hat{w}_l \cdot \left(1 + \sum_{j \in \Omega^{+,l}} H(s_j - s_k)\right)}{\text{rank}(k)} \tag{40}$$

$$= \sum_{k \in \Omega^+} \frac{\sum_{l=1}^{L} \mathbb{1}(k,l) \cdot \hat{w}_l + \sum_{l=1}^{L} \sum_{j \in \Omega^{+,l}} \mathbb{1}(k,l) \cdot \hat{w}_l \cdot H(s_j - s_k)}{\text{rank}(k)} \tag{41}$$

$$= \sum_{k \in \Omega^+} \frac{\sum_{l=1}^{L} \mathbb{1}(k,l) \cdot \hat{w}_l + \sum_{l=1}^{L} \sum_{j \in \Omega^+} \mathbb{1}(j,l) \cdot \mathbb{1}(k,l) \cdot \hat{w}_l \cdot H(s_j - s_k)}{\text{rank}(k)} \tag{42}$$

$$= \sum_{k \in \Omega^+} \frac{\sum_{l=1}^{L} \mathbb{1}(k,l) \cdot \hat{w}_l + \sum_{j \in \Omega^+} \sum_{l=1}^{L} \mathbb{1}(j,l) \cdot \mathbb{1}(k,l) \cdot \hat{w}_l \cdot H(s_j - s_k)}{\text{rank}(k)} \tag{43}$$

We define the following relevance function:

$$\text{rel}(k) = \sum_{l=1}^{L} \mathbb{1}(k,l) \cdot \hat{w}_l \tag{44}$$

By construction of $\mathbb{1}(\cdot,l)$:

$$\sum_{l=1}^{L} \mathbb{1}(j,l) \cdot \mathbb{1}(k,l) \cdot \hat{w}_l = \min(\text{rel}(k), \text{rel}(j)) \tag{45}$$

Using the definition of the relevance function from Eq. (44) and Eq. (45), we can rewrite Eq. (43) with $\mathcal{H}$-rank$^+$:

$$\Sigma w\text{AP} = \sum_{k \in \Omega^+} \frac{\text{rel}(k) + \sum_{j \in \Omega^+} \min(\text{rel}(j), \text{rel}(k)) \cdot H(s_j - s_k)}{\text{rank}(k)} \tag{46}$$

$$= \sum_{k \in \Omega^+} \frac{\mathcal{H}\text{-rank}^+(k)}{\text{rank}(k)} \tag{47}$$

Eq. (47) lacks the normalization constant $\sum_{k \in \Omega^+} \text{rel}(k)$ in order to have the same shape as $\mathcal{H}$-AP in Eq. (29). So we must prove that $\sum_{k \in \Omega^+} \text{rel}(k) = 1$:

$$\sum_{k \in \Omega^+} \text{rel}(k) = \sum_{k \in \Omega^+} \sum_{l=1}^{L} \mathbb{1}(k,l) \cdot \hat{w}_l \tag{48}$$

$$= \sum_{l=1}^{L} |\Omega^{(l)}| \sum_{p=1}^{l} \hat{w}_p \tag{49}$$

$$= \sum_{l=1}^{L} |\Omega^{(l)}| \sum_{p=1}^{l} \frac{w_p}{|\Omega^{+,p}|} \tag{50}$$

$$= \sum_{l=1}^{L} |\Omega^{(l)}| \sum_{p=1}^{l} \frac{w_p}{|\bigcup_{q=p}^{L} \Omega^{(q)}|} \tag{51}$$

$$= \sum_{l=1}^{L} |\Omega^{(l)}| \sum_{p=1}^{l} \frac{w_p}{\sum_{q=p}^{L} |\Omega^{(q)}|} \tag{52}$$

$$= \sum_{l=1}^{L} \sum_{p=1}^{l} \frac{|\Omega^{(l)}| \cdot w_p}{\sum_{q=p}^{L} |\Omega^{(q)}|} \tag{53}$$

$$= \sum_{p=1}^{L} \sum_{l=p}^{L} \frac{|\Omega^{(l)}| \cdot w_p}{\sum_{q=p}^{L} |\Omega^{(q)}|} \tag{54}$$

$$= \sum_{p=1}^{L} w_p \cdot \frac{\sum_{l=p}^{L} |\Omega^{(l)}|}{\sum_{q=p}^{L} |\Omega^{(q)}|} \tag{55}$$

$$= \sum_{p=1}^{L} w_p = 1 \tag{56}$$

We have proved that $\Sigma w\text{AP} = \mathcal{H}$-AP with the relevance function of Eq. (44):

$$\Sigma w\text{AP} = \frac{1}{\sum_{k \in \Omega^+} \text{rel}(k)} \sum_{k \in \Omega^+} \frac{\mathcal{H}\text{-rank}^+(k)}{\text{rank}(k)} = \mathcal{H}\text{-AP} \tag{57}$$

Finally we show, for an instance $k \in \Omega^{(l)}$, :

$$\text{rel}(k) = \sum_{p=1}^{L} \mathbb{1}(k,p) \cdot \hat{w}_p = \sum_{p=1}^{l} \cdot \hat{w}_p = \sum_{p=1}^{l} \frac{w_p}{|\Omega^{+,p}|} \tag{58}$$

i.e. the relevance of Eq. (44) is the same as the relevance of Property 1. This concludes the proof of Property 1. □

APPENDIX C
ADDITIONAL EXPERIMENTAL RESULTS

A. Additional hierarchical results

1) ASI.: The ASI [81] measures at each rank $n \leq N$ the set intersection proportion (SI) between the ranked list $a_1,...,a_N$ and the ground truth ranking $b_1,...,b_N$, with N the total number of positives. As it compares intersection the ASI can naturally take into account the different levels of semantic:

$$SI(n) = \frac{|\{a_1,...,a_n\} \cap \{b_1,...,b_n\}|}{n}$$

$$ASI = \frac{1}{N} \sum_{n=1}^{N} SI(n)$$

TABLE VIII: Performance comparison on Dynamic Metric Learning benchmarks [33].

	Method	DyML-Vehicle			DyML-Animal			DyML-Product		
		mAP	ASI	R@1	mAP	ASI	R@1	mAP	ASI	R@1
Fine	TL_{SH} [5]	26.1	38.6	84.0	37.5	46.3	66.3	36.32	46.1	59.6
	NSM [11]	27.7	40.3	88.7	38.8	48.4	69.6	35.6	46.0	57.4
	Smooth-AP [27]	27.1	39.5	83.8	37.7	45.4	63.6	36.1	45.5	55.0
	ROADMAP [37]	27.1	39.6	84.5	34.4	42.6	62.8	34.6	44.6	62.5
Hier.	ΣTL_{SH} [5]	25.5	38.1	81.0	38.9	47.2	65.9	36.9	46.3	58.5
	ΣNSM [11]	32.0	45.7	**89.4**	42.6	50.6	**70.0**	36.8	46.9	60.8
	CSL [33]	30.0	43.6	87.1	40.8	46.3	60.9	31.1	40.7	52.7
	CLCD-ACR [34]	16.0	42.9	-	36.0	57.1	-	29.4	58.8	-
	CLCD-ICR [34]	16.6	43.7	-	35.7	56.0	-	30.2	59.5	-
	HAPPIER	**37.0**	**49.8**	89.1	**43.8**	**50.8**	68.9	**38.0**	**47.9**	**63.7**

TABLE IX: Comparison of HAPPIER *v.s.* methods trained only on fine-grained labels on SOP and iNat-base. Metrics are reported for both semantic levels.

		SOP			iNat-base		
		Fine		Coarse	Fine		Coarse
	Method	R@1	AP	AP	R@1	AP	AP
Fine	TL_{SH} [5]	79.8	59.6	14.5	66.3	33.3	51.5
	NSM [11]	81.3	61.3	13.4	70.2	37.6	38.8
	NCA++ [15]	81.4	61.7	13.6	67.3	37.0	44.5
	Smooth-AP [27]	81.3	61.7	13.4	67.3	35.2	53.1
	ROADMAP [37]	**82.2**	**62.5**	12.9	69.3	35.1	50.4
Hier.	CSL [33]	79.4	58.0	45.0	62.9	30.2	88.5
	HAPPIER	81.0	60.4	**58.4**	70.7	36.7	**88.6**
	$HAPPIER_F$	81.8	62.2	36.0	**71.6**	**37.8**	85.1

TABLE X: Impact of optimization choices for $\mathcal{H}$-AP (cf. Sec. III-B) on iNat-base.

$\mathcal{L}^s_{\mathcal{H}\text{-AP}}$	$\mathcal{L}^*_{\text{DG}}$	$\mathcal{H}$-AP
✗	✗	52.3
✓	✗	53.1
✓	✓	**54.3**

TABLE XI: Comparison of $\mathcal{H}$-AP (Eq. (20)) and ΣwAP from Property 1.

test→ train↓	$\mathcal{H}$-AP	$\sum w$AP	NDCG
$\mathcal{H}$-AP	**53.1**	39.8	**97.0**
$\sum w$AP	52.0	**40.5**	96.4

2) Dynamic metric learning results.: On Tab. VIII, we evaluate HAPPIER on the recent DyML benchmarks [33]. HAPPIER again shows significant gains in mAP and ASI compared to methods only trained on fine-grained labels, *e.g.* +9pt in mAP and +10pt in ASI on DyML-V. HAPPIER also outperforms other hierarchical baselines: +4.8pt mAP on DyML-V, +0.9 on DyML-A and +1.8 on DyML-P. In R@1, HAPPIER performs on par with other methods on DyML-V and outperforms other hierarchical baselines by a large margin on DyML-P: 63.7 *v.s.* 60.8 for ΣNSM. Interestingly, HAPPIER also consistently outperforms CSL [33] on its own datasets[1].

3) Detailed evaluation: Tab. IX shows the different methods' performances on all semantic hierarchy levels. We evaluate HAPPIER and $HAPPIER_F$ with $\alpha = 5$ on SOP and $\alpha = 3$ on iNat-base. Similarly to Tab. VI, Tab. IX shows that HAPPIER gives the best performances at the coarse level, with a significant boost compared to fine-grained methods, *e.g.* +43.9pt AP compared to the best non-hierarchical TL_{SH} [5] on SOP. HAPPIER even outperforms the best fine-grained methods in R@1 on iNat-base, but is slightly below on SOP. $HAPPIER_F$ performs on par with the best methods at the finest level on SOP, while further improving performances on iNat-base, and still significantly outperforms fine-grained methods at the coarse level.

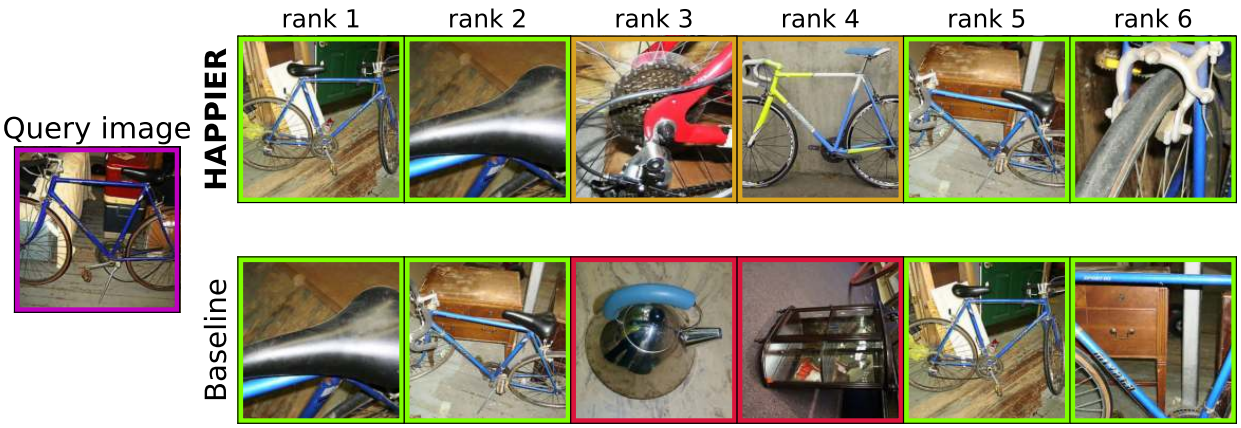
4) Relevance function choice: Tab. XI compares models that are trained with the relevance function of Eq. (20), *i.e.* $\mathcal{H}$-AP, and $\sum w$AP (relevance given in Property 1). We report results for $\mathcal{H}$-AP, $\sum w$AP and NDCG. Both $\mathcal{H}$-AP, $\sum w$AP perform better when trained with their own metric: +1.1pt $\mathcal{H}$-AP for the model trained to optimize it and +0.7pt $\sum w$AP for the model trained to optimize it. Both models show similar performances in NDCG (96.4 *v.s.* 97.0).

5) Choices of optimization: In Tab. X, we study the impact of our different choices regarding the direct optimization of $\mathcal{H}$-AP. The baseline method uses a sigmoid to optimize $\mathcal{H}$-AP as in [27], [51]. Switching to our surrogate loss $\mathcal{L}^s_{\mathcal{H}\text{-AP}}$ Sec. III-B yields a +0.8pt increase in $\mathcal{H}$-AP. Finally, the combination with $\mathcal{L}_{\text{DG}}$ in HAPPIER results in an additional 1.3pt improvement in $\mathcal{H}$-AP.

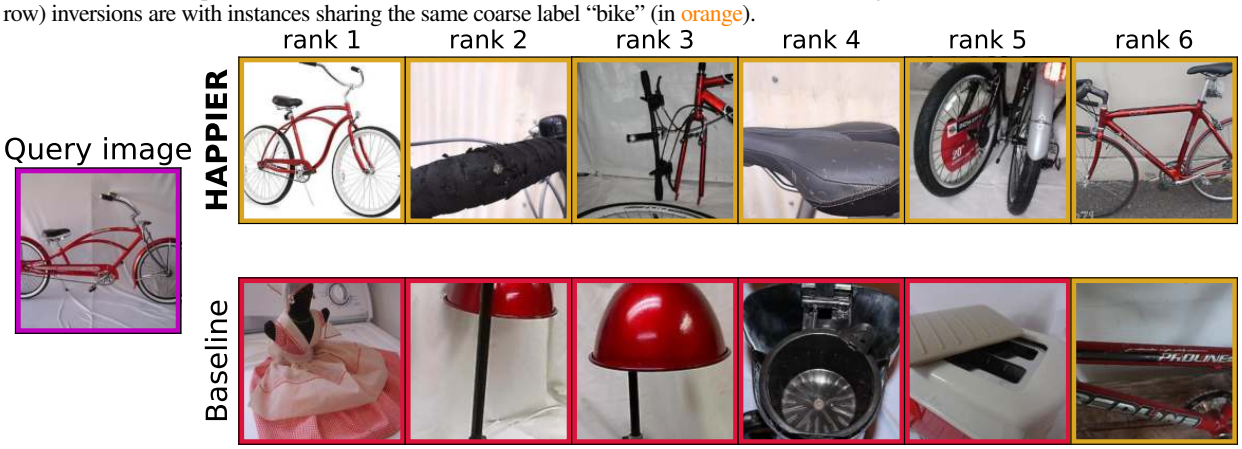
B. Additional qualitative results

Controlled errors: SOP We showcase in Fig. 15 errors of HAPPIER *v.s.* a fine-grained baseline. On Fig. 15a, we illustrate how a model trained with HAPPIER makes mistakes that are less severe than a baseline model trained only on the fine-grained level. On Fig. 15b, we show an example where both models fail

[1]CSL's score on Tab. VIII are above those reported in [33]; personal discussions with the authors [33] validate that our results are valid for CSL.



(a) HAPPIER can help make less severe mistakes. The inversion on the bottom row are with negative instances (in red), where as with HAPPIER (top row) inversions are with instances sharing the same coarse label "bike" (in orange).



(b) In this example, the models fail to retrieve the correct fine grained images. However HAPPIER still retrieves images of very similar bikes (in orange) whereas the baseline retrieves images that are dissimilar semantically to the query (in red).

Fig. 15: Qualitative examples of failure cases from a standard fine-grained model corrected by training with HAPPIER.

to retrieve the correct fine-grained instances, however the model trained with HAPPIER retrieves images of bikes that are visually more similar to the query.