Hierarchical Average Precision Training for Pertinent Image Retrieval

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

Hierarchical Average Precision Training for Pertinent Image Retrieval

Elias Ramzi[1,2], Nicolas Audebert[1], Nicolas Thome[1,3], Clément Rambour[1] and Xavier Bitot[2]

[1] CEDRIC, Conservatoire National des Arts et Métiers, Paris, France
{elias.ramzi,nicolas.audebert,nicolas.thome,clement.rambour}@cnam.fr
[2] Coexya, Paris, France
xavier.bitot@coexya.eu
[3] Sorbonne Université, CNRS, ISIR, F-75005 Paris, France

Abstract. Image Retrieval is commonly evaluated with Average Precision (AP) or Recall@k. Yet, those metrics, are limited to binary labels and do not take into account errors' severity. This paper introduces a new hierarchical AP training method for pertinent image retrieval (HAPPIER). HAPPIER is based on a new $\mathcal{H}$-AP metric, which leverages a concept hierarchy to refine AP by integrating errors' importance and better evaluate rankings. To train deep models with $\mathcal{H}$-AP, we carefully study the problem's structure and design a smooth lower bound surrogate combined with a clustering loss that ensures consistent ordering. Extensive experiments on 6 datasets show that HAPPIER significantly outperforms state-of-the-art methods for hierarchical retrieval, while being on par with the latest approaches when evaluating fine-grained ranking performances. Finally, we show that HAPPIER leads to better organization of the embedding space, and prevents most severe failure cases of non-hierarchical methods. Our code is publicly available at https://github.com/elias-ramzi/HAPPIER.

Keywords: Hierarchical Image Retrieval, Hierarchical Average Precision, Ranking

1 Introduction

Image Retrieval (IR) consists in ranking images with respect to a query by decreasing order of visual similarity. IR methods are commonly evaluated using Recall@k (R@k) or Average Precision (AP). Because those metrics are non-differentiable, a rich literature exists on finding adequate surrogate loss functions to optimize them with deep learning, with tuple-wise losses [40,46,57,54,55], proxy based losses [59,53,12,49] and direct AP optimization methods [6,43,34,45,2,42].

These metrics are only defined for binary ($\oplus$/$\ominus$) labels, which we denote as *fine-grained labels*: an image is negative as soon as it is not strictly similar to the query. Binary metrics are by design unable to take into account the severity of the mistakes in a ranking. On Fig. 1, some negative instances are "less negative" than others, *e.g.* given the "Brown Bear" query, "Polar bear" is more relevant than

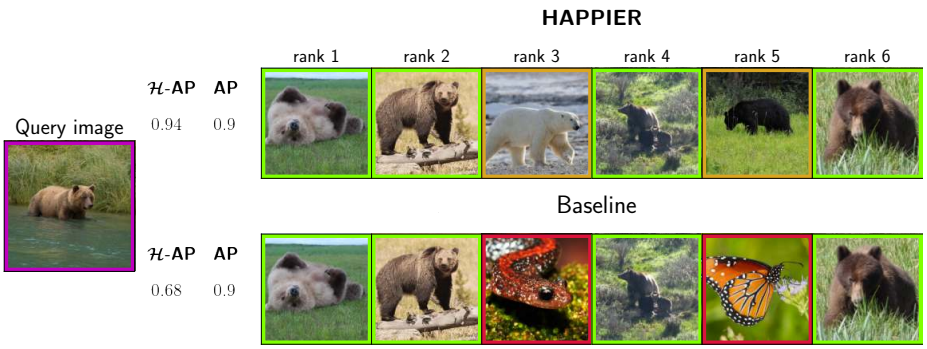


Fig. 1: **Proposed HAPPIER framework for pertinent image retrieval**. Standard ranking metrics based on binary labels, *e.g.* Average Precision (AP), assign the same score to the bottom and top row rankings (0.9). We introduce the $\mathcal{H}$-AP metric based on non-binary labels, that takes into account mistakes' severity. $\mathcal{H}$-AP assigns a smaller score to the bottom row (0.68) than the top one (0.94). HAPPIER maximizes $\mathcal{H}$-AP during training and thus explicitly supports to learn rankings similar to the top one, in contrast to binary ranking losses.

"Butterfly". However, AP is 0.9 for both the top and bottom rankings. Consequently, training on binary metrics (*e.g.* AP or R@k) develops no incentive to produce ranking such as the top row, and often produces rankings similar to the bottom one. To address this problem, we introduce the HAPPIER method dedicated to Hierarchical Average Precision training for Pertinent ImagE Retrieval. HAPPIER provides a smooth training objective, amenable to gradient descent, which explicitly takes into account the severity of mistakes when evaluating rankings.

Our first contribution is to define a new Hierarchical AP metric ($\mathcal{H}$-AP) that leverages the hierarchical tree between concepts and enables a fine weighting between errors in rankings. As shown in Fig. 1, $\mathcal{H}$-AP assigns a larger score (0.94) to the top ranking than to the bottom one (0.68). We show that $\mathcal{H}$-AP provides a consistent generalization of AP for the non-binary setting. We also introduce our HAPPIER$_\text{F}$ variant, giving more weights to fine-grained levels of the hierarchy. Since $\mathcal{H}$-AP, like AP, is a non-differentiable metric, our second contribution is to use HAPPIER to directly optimize $\mathcal{H}$-AP by gradient descent. We carefully design a smooth surrogate loss for $\mathcal{H}$-AP that has strong theoretical guarantees and is an upper bound of the true loss. We then define an additional clustering loss to support having a consistency between partial and global rankings.

We validate HAPPIER on six IR datasets, including three standard datasets (Stanford Online Products [33] and iNaturalist-base/full [51]), and three recent hierarchical datasets (DyML [47]). We show that, when evaluating on hierarchical metrics (*e.g.* $\mathcal{H}$-AP), HAPPIER outperforms state-of-the-art methods for fine-grained ranking [57,59,49,42], the baselines and the latest hierarchical method of [47], and only slightly under-performs *vs.* state-of-the-art IR methods at the fine-grained level (*e.g.* AP, R@1). HAPPIER$_\text{F}$ performs on par on fine-grained metrics while still outperforming fine-grained methods on hierarchical metrics.

2 Related work

2.1 Image Retrieval and ranking

The Image Retrieval community has designed several families of methods to optimize metrics such as AP and R@k. Methods that relies on tuplet-wise losses, like pair losses [17,41], triplet losses [57], or larger tuplets [46,27,54] learn comparison relations between instances. Methods using proxies have been introduced to lower the computational complexity of tuplet based training [31,59,53,12,49]: 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 neighbourhood. Finally, there also has been large amounts of work dedicated to the direct optimization of the AP during training by introducing differentiable surrogates [6,43,34,45,2,42], so that models are optimized on the same metric they are evaluated on. However, nearly all of these methods only consider binary labels: two instances are either the same (positive) or different (negative), leading to poor performance when multiple levels of hierarchy are considered.

2.2 Hierarchical predictions and metrics

There has been a recent regain of interest in Hierarchical Classification [13,1,8] with the introductions of methods based either on a hierarchical softmax function or on multiple classifiers. It is considered that learning from hierarchical relations between labels leads to more robust models that make "better mistakes" [1]. Yet, hierarchical classification means that labels are known in advance and are identical in the train and test sets. This is called a *closed set* setting. However, Hierarchical Image Retrieval does not fall into this framework. Standard IR protocols consider the *open set* paradigm to better evaluate the generalization abilities of learned models: the retrieval task at test time pertains to labels that were not present in the train set, making classification poorly suited to IR.

Meanwhile, the broader Information Retrieval community has been using datasets where documents can be more or less relevant depending on the query and the user making the request [20,24]. Instead of the mere positive/negative dichotomy, each instance has a continuous score quantifying its relevance to the query. To quantify the quality of their retrieval engine, Information Retrieval researchers have long used ranking based metrics, such as the NDCG [22,10], that penalize mistakes differently based on whether they occur at the top or the bottom of the ranking and whether wrong documents still have some marginal relevance or not. Average Precision is also used as a retrieval metric [23] and has even been given probabilistic interpretations based on how users interact with the system [14]. Several works have investigated how to optimize those metrics during the training of neural networks, *e.g.* using pairwise losses [4] and later using smooth surrogates of the NDCG in LambdaRank [5], SoftRank [48], ApproxNDCG [38] and Learning-To-Rank [3]. These works however focused on NDCG, the most popular metric for information retrieval, 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 of this literature is that NDCG does not relate easily to average precision [15], which is 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 [44,14]. The graded Average Precision from [44] is the closest to our goal as it leverages SoftRank for direct optimization on non-binary relevance judgements, 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 unpractical for large scale settings and was only applied to small-scale corpora of a few thousands documents, compared to the hundred thousands of images in IR.

Recently, the authors of [47] 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 and tries to structure the embedding space in a hierarchical manner. However, this method faces the same limitation as the usual 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 HAPPIER significantly outperforms CSL [47] both on hierarchical metrics and AP-level evaluations.

3 HAPPIER Model

We detail HAPPIER our Hierarchical Average Precision training method for Pertinent ImagE Retrieval. We first introduce the Hierarchical Average Precision, $\mathcal{H}$-AP in Sec. 3.1, that leverages a hierarchical tree (Fig. 2a) of labels. It is based on the hierarchical rank, $\mathcal{H}$-rank, and evaluates rankings so that more relevant instances are ranked before less relevant ones (Fig. 2b). We then show how to directly optimize $\mathcal{H}$-AP by stochastic gradient descent (SGD) using HAPPIER in Sec. 3.2. Our training objective combines a carefully designed smooth upper bound surrogate loss for $\mathcal{L}_{\mathcal{H}\text{-AP}} = 1 - \mathcal{H}\text{-AP}$ and a clustering loss $\mathcal{L}_{\text{clust.}}$ that supports consistent rankings.

Context Let us consider a retrieval set $\Omega = \{\boldsymbol{x_j}\}_{j \in [\![1;N]\!]}$ composed of N instances. For a query[1] $\boldsymbol{q} \in \Omega$, we aim to order all $x_j \in \Omega$ so that more relevant (*i.e.* similar) instances are ranked before less relevant instances.

In our hierarchical setting, the relevance of an instance $\boldsymbol{x_j}$ is non-binary. We assume that we have access to a hierarchical tree defining semantic similarities between concepts as in Fig. 2a. For a query $\boldsymbol{q}$, we leverage this knowledge to 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): for $L=3$ and a "Lada #2" query, $\Omega^{(3)}$ are the images of the same "Lada #2" (green), see Fig. 2. The set $\Omega^{(l)}$ for $l < L$ contains instances with smaller

[1] For the sake of readability, our notations are given for a single query. During training, HAPPIER optimizes our hierarchical retrieval objective by averaging several queries.

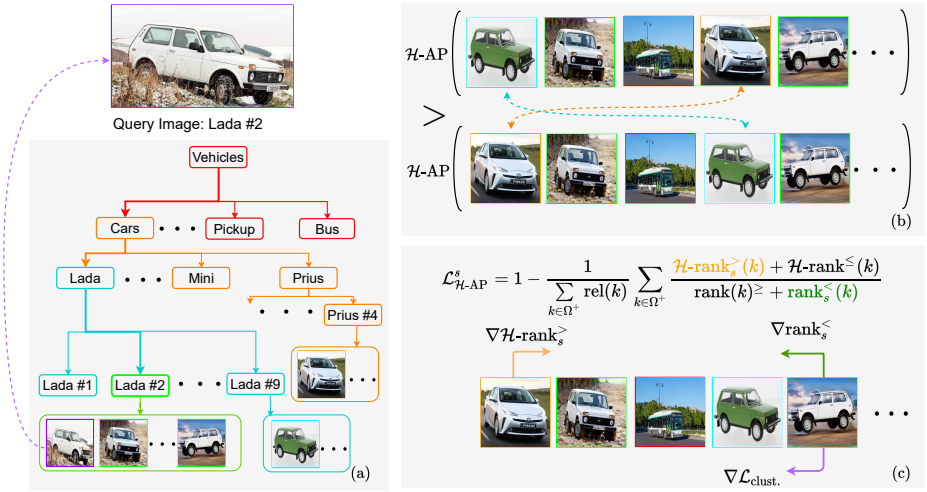


Fig. 2: HAPPIER leverages a hierarchical tree representing the semantic similarities between concepts in (a) to introduce a new hierarchical metric, $\mathcal{H}$-AP in Eq. (3), see (b). $\mathcal{H}$-AP exploits the hierarchy to weight rankings' inversion: given the query image of a "Lada #2", $\mathcal{H}$-AP penalizes an inversion with a "Lada #9" less than with a "Prius #4". To directly train models with $\mathcal{H}$-AP, we carefully study the structure of the problem and introduce the $\mathcal{L}^s_{\mathcal{H}\text{-AP}}$ loss in Eq. (5), which provides a smooth upper bound of $\mathcal{L}_{\mathcal{H}\text{-AP}}$, see (c). We also train HAPPIER with the $\mathcal{L}_{\text{clust.}}$ loss in Eq. (6) to enforce the partial ordering in stochastic optimization to mach the global ones.

relevance with respect to the query: $\Omega^{(2)}$ in Fig. 2 is the set of "Lada" that are not "Lada #2" (blue) and $\Omega^{(1)}$ is the set of "Cars" that are not "Lada" (orange). We also define $\Omega^- := \Omega^{(0)}$ as the set of negative instances, *i.e.* the set of vehicles that are not "Cars" (in red) in Fig. 2 and $\Omega^+ = \bigcup_{l=1}^{L} \Omega^{(l)}$. Each instance k of $\Omega^{(l)}$ is thus associated a value through the *relevance function* denoted as $\mathbf{rel(}\boldsymbol{k}\mathbf{)}$ [20].

To rank the instances $x_j \in \Omega$ with respect to the query $\boldsymbol{q}$, we compute cosine similarities in an embedding space. More precisely, we extract embedding vectors using a deep neural network $\boldsymbol{f}$ parameterized by $\boldsymbol{\theta}$, $v_j = f_\theta(x_j)$, and compute the cosine similarity between the query and every image $s_j = f_\theta(q)^T v_j$. Images are then ranked by decreasing cosine similarity score. We learn the parameters $\boldsymbol{\theta}$ of the network with HAPPIER, our framework to directly minimize $\mathcal{L}_{\mathcal{H}\text{-AP}}(\theta) = 1 - \mathcal{H}\text{-AP}(\theta)$. This enforces a ranking where the instances with the highest cosine similarity scores belong to $\Omega^{(L)}$, then $\Omega^{(L-1)}$ *etc.* and the items with the lowest cosine similarity belong to Ω^-.

3.1 Hierarchical Average Precision

Average Precision (AP) is the most common metric in Image Retrieval. AP evaluates a ranking in a binary setting: for a given query, each instance is either

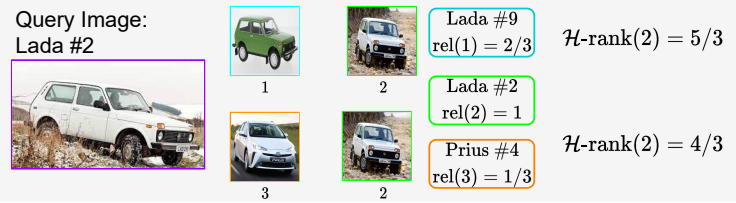


Fig. 3: Given a "Lada #2" query, the top inversion is less severe than the bottom one. Indeed on the top row instance 1 is semantically closer to the query – as it is a "Lada"– than instance 3 on the bottom row. Indeed instance 3's closest common ancestor with the query, "Cars", is farther in the hierarchical tree (see Fig. 2a). Because of that $\mathcal{H}$-rank(2) is greater on the top row (5/3) than on the bottom row (4/3), leading to a greater $\mathcal{H}$-AP in Fig. 2b for the top row.

positive or negative. It is computed as the average of precision at each rank n over the positive set $\text{AP} = \frac{1}{|\Omega^+|}\sum_{n=1}^{N}\text{Prec}(n)$. Previous works have written the AP using the ranking operator [2] as in Eq. (1). The rank for an instance k is written as a sum of Heaviside (step) function H [38]: this counts the number of instances j ranked before k, *i.e.* that have a higher cosine similarity ($s_j > s_k$). rank^+ is the rank among the positive instances, *i.e.* restricted to Ω^+.

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

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

$$\mathcal{H}\text{-rank}(k) = \text{rel}(k) + \sum_{j\in\Omega^+}\min(\text{rel}(k), \text{rel}(j))\cdot H(s_j - s_k)\ . \tag{2}$$

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. 3, $\mathcal{H}$-rank induces a smoother penalization for instances that do not share the same fine-grained label as the query but still share some coarser semantics, which is not the case for rank^+.

From $\mathcal{H}$-rank in Eq. (2) we define the Hierarchical Average Precision, $\mathcal{H}$-AP:

$$\mathcal{H}\text{-AP} = \frac{1}{\sum_{k\in\Omega^+}\text{rel}(k)}\sum_{k\in\Omega^+}\frac{\mathcal{H}\text{-rank}(k)}{\text{rank}(k)} \tag{3}$$

Eq. (3) extends the AP to non-binary labels. We replace rank^+ by our hierarchical rank $\mathcal{H}$-rank and the normalization term $|\Omega^+|$ is replaced by $\sum_{k\in\Omega^+}\text{rel}(k)$, which both represent the "sum of positives", see more details in supplementary A.2.

$\mathcal{H}$-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 supplementary A.2).

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 [37,9], 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 $\Omega^{(l)}$:

$$\text{rel}(k) = \frac{(l/L)^\alpha}{|\Omega^{(l)}|} \text{ if } k \in \Omega^{(l)} \tag{4}$$

Other definitions fulfilling the decreasing similarity behaviour in the tree are possible. 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 A.2), *i.e.* the weighted sum of AP is a particular case of $\mathcal{H}$-AP.

We set $\alpha = 1$ in Eq. (4) 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 denoted HAPPIER_F and discussed in Sec. 4.

3.2 Direct optimization of $\mathcal{H}$-AP

$\mathcal{H}$-AP in Eq. (3) involves the computation of $\mathcal{H}$-rank and rank, which are non-differentiable due to the summing of Heaviside step functions. We thus introduce a smooth approximation of $\mathcal{H}$-AP to obtain a surrogate loss amenable to gradient descent, which fulfils theoretical guarantees for proper optimization.

Re-writing $\mathcal{H}$-AP In order to design our surrogate loss for $\mathcal{L}_{\mathcal{H}\text{-AP}} = 1 - \mathcal{H}\text{-AP}$, we decompose $\mathcal{H}$-rank and rank into two quantities. Denoting $\mathcal{H}\text{-rank}^{>}(k)$ (resp. $\mathcal{H}\text{-rank}^{\leq}(k)$) as the restriction of $\mathcal{H}$-rank to instances of strictly higher relevances (resp. lower or equal), we can see that $\mathcal{H}\text{-rank}(k) = \mathcal{H}\text{-rank}^{>}(k) + \mathcal{H}\text{-rank}^{\leq}(k)$. The rank can be decomposed in a similar fashion: $\text{rank}(k) = \text{rank}^{\geq}(k) + \text{rank}^{<}(k)$ where $<$ (resp. $\geq$) denotes the restriction to instances of strictly lower relevances (resp. higher or equal). The $\mathcal{L}_{\mathcal{H}\text{-AP}}$ can be rewritten as follow:

$$\mathcal{L}_{\mathcal{H}\text{-AP}} = 1 - \frac{1}{\sum_{k \in \Omega^+} \text{rel}(k)} \sum_{k \in \Omega^+} \frac{\mathcal{H}\text{-rank}^{>}(k) + \mathcal{H}\text{-rank}^{\leq}(k)}{\text{rank}^{\geq}(k) + \text{rank}^{<}(k)} \ . \tag{5}$$

We choose to optimize over $\mathcal{H}$-rank$^{>}$ and rank$^{<}$ in Eq. (5). We maximize $\mathcal{H}$-rank$^{>}$ to enforce that the k^{th} instance must decrease in cosine similarity score if it is ranked before another instance of higher relevance ($\nabla\mathcal{H}$-rank$^{>}$ in Fig. 2 enforces the blue instance to be ranked after the green one as it is less relevant to the query). We minimize rank$^{<}$ to encourage the k^{th} instance to increase in cosine similarity score if it is ranked after one or more instances of lower relevance (∇rank$^{<}$ in Fig. 2 enforces that the last green instance moves before less relevant instances). Optimizing both those terms leads to a decrease in $\mathcal{L}_{\mathcal{H}\text{-AP}}$. On the other hand, we purposely do not optimize the two remaining $\mathcal{H}$-rank$^{\leq}(k)$ and rank$^{\geq}(k)$ terms, since this could harm training performances as explained in supplementary A.3.

Upper bound of $\mathcal{L}_{\mathcal{H}\text{-AP}}$ Based on the previous analysis, we now design our surrogate loss $\mathcal{L}^{s}_{\mathcal{H}\text{-AP}}$ by introducing a smooth approximation of rank$^{<}$ and $\mathcal{H}$-rank$^{>}(k)$. An important sought property of $\mathcal{L}^{s}_{\mathcal{H}\text{-AP}}$ is that it is an upper bound of $\mathcal{L}_{\mathcal{H}\text{-AP}}$. To this end, we approximate $\mathcal{H}$-rank$^{>}(k)$ with a piece-wise linear function that is a lower bound of the Heaviside function. rank$^{<}$ is approximated with a smooth upper bound of the Heaviside that combines a piece-wise sigmoid function and an affine function, which has been shown to make the training more robust thanks to the induced implicit margins between positives and negatives [45,2,42]. More details are given in supplementary A.3 on those surrogates.

Clustering constraint in HAPPIER Positives only need to have a greater cosine similarity with the query than negatives in order to be correctly ranked. Yet, we cannot optimize the ranking on the entire datasets – and thus the true $\mathcal{L}_{\mathcal{H}\text{-AP}}$ – because of the batch-wise estimation performed in stochastic gradient descent. To mitigate this issue, we take inspiration from clustering methods [59,49] to define the following objective in order to group closely the embeddings of instances that share the same fine-grained label:

$$\mathcal{L}_{\text{clust.}}(\theta) = -\log\left(\frac{\exp(\frac{v_y^T p_y}{\sigma})}{\sum_{p_z\in\mathcal{Z}}\exp(\frac{v_y^T p_z}{\sigma})}\right), \tag{6}$$

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 σ is a temperature scaling parameter. In Fig. 2, $\nabla\mathcal{L}_{\text{clust.}}$ further clusters "Lada #2" instances. $\mathcal{L}_{\text{clust.}}$ induces a reference shared across batches and thus enforces that the partial ordering in-between batches is consistent with the global ordering over the entire retrieval set.

Our resulting final objective is a linear combination of both our losses, with a weight factor $\lambda\in[0,1]$ that balances the two terms:

$$\mathcal{L}_{\text{HAPPIER}}(\theta) = (1-\lambda)\cdot\mathcal{L}^{s}_{\mathcal{H}\text{-AP}}(\theta) + \lambda\cdot\mathcal{L}_{\text{clust.}}(\theta)\ .$$

4 Experiments

4.1 Experimental setup

Datasets We use the standard benchmark Stanford Online Products [33] (SOP) with two levels of hierarchy ($L = 2$), and iNaturalist-2018 [51] with the standard splits from [2] in two settings: i) iNat-base with two levels of hierarchy ($L=2$) ii) iNat-full with the full biological taxonomy composed of 7 levels ($L=7$). We also evaluate on the recent dynamic metric learning (DyML) datasets (DyML-V, DyML-A, DyML-P) introduced in [47] for the task of hierarchical image retrieval, each with 3 semantic levels ($L=3$).

Implementation details Our base model is a ResNet-50 pretrained on ImageNet for SOP and iNat-base/full, and a ResNet-34 randomly initialized on DyML-V&A and pretrained on ImageNet on DyML-P, following [47]. Unless specified otherwise, all reported results are obtained with $\alpha=1$ in Eq. (4) and $\lambda=0.1$ for $\mathcal{L}_{\text{HAPPIER}}$. We study the impact of these parameters in Sec. 4.3.

Metrics For SOP and iNat, we evaluate the models based on three hierarchical metrics: $\mathcal{H}$-AP – which we introduced in Eq. (3) – the Average Set Intersection (ASI) and the Normalized Discounted Cumulative Gain (NDCG), defined in supplementary B.3. We also report the AP for each semantic level. For DyML, we follow the evaluation protocols of [47] and compute AP, ASI and R@1 on each semantic scale before averaging them. We cannot compute $\mathcal{H}$-AP or NDCG on those datasets as the hierarchical tree is not available on the test set.

Baselines We compare HAPPIER to several recent image retrieval methods optimized at the fine-grained level, which represent strong baselines for IR when training with binary labels: Triplet SH (TL_{SH}) [57], NormSoftMax (NSM) [59], ProxyNCA++ (NCA++) [49] and ROADMAP [42]. We also benchmark against hierarchical methods obtained by summing these fine-grained losses at different levels (denoted by Σ), and with respect to the recent hierarchical CSL loss [47]. Details on the experimental setup are given in supplementary B.

4.2 Main Results

Hierachical results We first evaluate HAPPIER on global hierarchical metrics. On Tab. 1, we notice 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 +16.1pt on SOP, +13pt on iNat-base and 12.7pt 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 [47], 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.

Table 1: Comparison of HAPPIER on SOP and iNat-base/full when using hierarchical metrics. Best results in **bold**, second best underlined.

	Method	SOP			iNat-base			iNat-full		
		$\mathcal{H}$-AP	ASI	NDCG	$\mathcal{H}$-AP	ASI	NDCG	$\mathcal{H}$-AP	ASI	NDCG
Fine	Triplet SH [57]	42.2	22.4	78.8	39.5	63.7	91.5	36.1	59.2	89.8
Fine	NSM [59]	42.8	21.1	78.3	38.0	51.6	88.9	33.3	51.7	88.2
Fine	NCA++ [49]	43.0	21.5	78.4	39.5	57.0	90.1	35.3	55.7	89.0
Fine	Smooth-AP [2]	42.9	20.6	78.2	41.3	64.2	91.9	37.2	60.1	90.1
Fine	ROADMAP [42]	43.3	19.1	77.9	40.3	61.0	91.2	34.7	59.6	89.5
Hier.	ΣTL$_{\text{SH}}$ [57]	53.1	53.3	89.2	44.0	87.4	96.4	39.9	85.5	92.0
Hier.	ΣNSM [59]	50.4	49.7	87.0	47.9	75.8	94.4	46.9	74.2	**93.8**
Hier.	ΣNCA++ [49]	49.5	52.8	87.8	48.9	78.7	95.0	44.7	74.3	92.6
Hier.	CSL [47]	52.8	57.9	88.1	50.1	**89.3**	96.7	45.1	84.9	93.0
	HAPPIER	**59.4**	**65.9**	**91.5**	**54.3**	**89.3**	**96.9**	**47.9**	**87.2**	**93.8**

On Tab. 2, we evaluate HAPPIER on the recent DyML benchmarks. 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 *vs.* 60.8 for ΣNSM. Interestingly, HAPPIER also consistently outperforms CSL [47] on its own datasets[2].

Table 2: Performance comparison on Dynamic Metric Learning benchmarks [47].

	Method	DyML-Vehicle			DyML-Animal			DyML-Product		
		mAP	ASI	R@1	mAP	ASI	R@1	mAP	ASI	R@1
Fine	TL$_{\text{SH}}$ [57]	26.1	38.6	84.0	37.5	46.3	66.3	36.32	46.1	59.6
Fine	NSM [59]	27.7	40.3	88.7	38.8	48.4	69.6	35.6	46.0	57.4
Fine	Smooth-AP [2]	27.1	39.5	83.8	37.7	45.4	63.6	36.1	45.5	55.0
Fine	ROADMAP [42]	27.1	39.6	84.5	34.4	42.6	62.8	34.6	44.6	62.5
Hier.	ΣTL$_{\text{SH}}$ [57]	25.5	38.1	81.0	38.9	47.2	65.9	36.9	46.3	58.5
Hier.	ΣNSM [59]	32.0	45.7	**89.4**	42.6	50.6	**70.0**	36.8	46.9	60.8
Hier.	CSL [47]	30.0	43.6	87.1	40.8	46.3	60.9	31.1	40.7	52.7
	HAPPIER	**37.0**	**49.8**	89.1	**43.8**	**50.8**	68.9	**38.0**	**47.9**	**63.7**

[2] CSL's score on Tab. 2 are above those reported in [47]; personal discussions with the authors [47] validate that our results are valid for CSL, see supplementary B.5.

Detailed evaluation Tabs. 3 and 4 shows the different methods' performances on all semantic hierarchy levels. We evaluate HAPPIER and also HAPPIER$_{\text{F}}$ ($\alpha > 1$ for Eq. (4) in Sec. 3.1), with $\alpha = 5$ on SOP and $\alpha = 3$ on iNat-base/full. HAPPIER optimizes the overall hierarchical performances, while HAPPIER$_{\text{F}}$ is meant to be optimal at the fine-grained level while still optimizing coarser levels.

Table 3: Comparison of HAPPIER *vs.* 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$_{\text{SH}}$ [57]	79.8	59.6	14.5	66.3	33.3	51.5
	NSM [59]	81.3	61.3	13.4	70.2	<u>37.6</u>	38.8
	NCA++ [49]	81.4	61.7	13.6	67.3	37.0	44.5
	Smooth-AP [2]	81.3	61.7	13.4	67.3	35.2	53.1
	ROADMAP [42]	**82.2**	**62.5**	12.9	69.3	35.1	50.4
Hier.	CSL [47]	79.4	58.0	<u>45.0</u>	62.9	30.2	<u>88.5</u>
	HAPPIER	81.0	60.4	**58.4**	<u>70.7</u>	36.7	**88.6**
	HAPPIER$_{\text{F}}$	<u>81.8</u>	<u>62.2</u>	36.0	**71.0**	**37.8**	78.8

On Tab. 3, we observe 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$_{\text{SH}}$ [57] on SOP. HAPPIER even outperforms the best fine-grained methods in R@1 on iNat-base, but is slightly below on SOP. HAPPIER$_{\text{F}}$ 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.

The satisfactory behaviour and the two optimal regimes of HAPPIER and HAPPIER$_{\text{F}}$ are confirmed and even more pronounced on iNat-full (Tab. 4): HAPPIER gives the best results on coarser levels (from "Order"), while being very close to the best results on finer ones. HAPPIER$_{\text{F}}$ gives the best results at the finest levels, even outperforming very competitive fine-grained baselines.

Again, note that HAPPIER outperforms CSL [47] on all semantic levels and datasets on Tabs. 3 and 4, *e.g.* +5pt on the fine-grained AP ("Species") and +3pt on the coarsest AP ("Kingdom") on Tab. 4.

4.3 HAPPIER analysis

Ablation study In Tab. 5, 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 [38,2]. Switching to our surrogate loss $\mathcal{L}^s_{\mathcal{H}\text{-AP}}$ Sec. 3.2 yields a +0.8pt increase in $\mathcal{H}$-AP. Finally, the combination with $\mathcal{L}_{\text{clust.}}$ in HAPPIER results in an additional 1.3pt improvement in $\mathcal{H}$-AP.

Table 4: Comparison of HAPPIER *vs.* methods trained only on fine-grained labels 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_{SH} [57]	66.3	33.3	34.2	32.3	35.4	48.5	54.6	68.4
	NSM [59]	70.2	**37.6**	38.0	31.4	28.6	36.6	43.9	63.0
	NCA++ [49]	67.3	37.0	37.9	33.0	32.3	41.9	48.4	66.1
	Smooth-AP [2]	67.3	35.2	36.3	33.5	35.0	49.3	55.8	69.9
	ROADMAP [42]	69.3	35.1	35.4	29.3	29.6	46.4	54.7	69.5
Hier.	CSL [47]	59.9	30.4	32.4	36.2	50.7	81.0	87.4	91.3
	HAPPIER	70.2	36.0	37.0	38.0	**51.9**	**81.3**	**89.1**	**94.4**
	$HAPPIER_F$	**70.8**	**37.6**	**38.2**	**38.8**	50.9	76.1	82.2	83.1

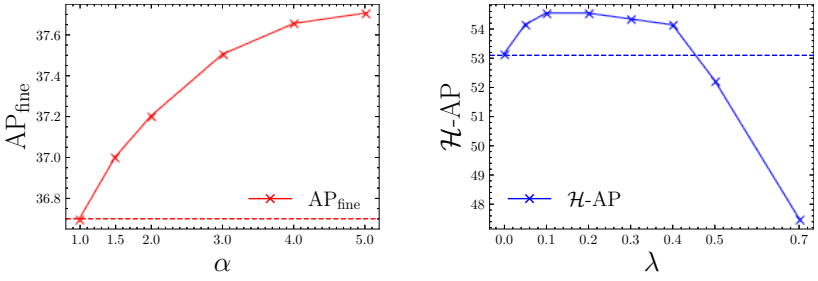
Table 5: Impact of optimization choices for $\mathcal{H}$-AP (cf. Sec. 3.2) on iNat-base.

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

Table 6: Comparison of $\mathcal{H}$-AP (Eq. (4)) and ΣwAP from supplementary A.2.

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

Impact of the relevance function Tab. 6 compares models that are trained with the relevance function of Eq. (4), *i.e.* $\mathcal{H}$-AP, and $\sum w$AP (relevance given in supplementary A.2). 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 *vs.* 97.0).



(a) AP_{fine} vs α in Eq. (4). (b) $\mathcal{H}$-AP *vs.* λ for $\mathcal{L}_{HAPPIER}$.

Fig. 4: Impact on Inat-base of α in Eq. (4) for setting the relevance of $\mathcal{H}$-AP (a) and of the λ hyper-parameter on HAPPIER results (b).

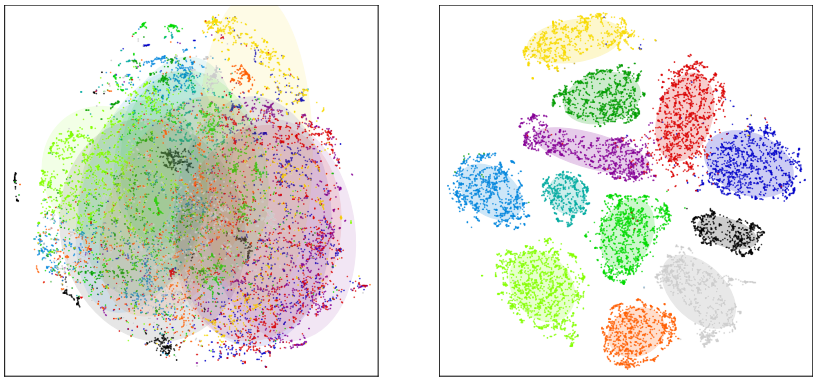
Hyper-parameters Fig. 4a studies the impact of α for setting the relevance in Eq. (4): increasing α improves the performances of the AP at the fine-grained level on iNat-base, as expected. We also show in Fig. 4b the impact of λ weighting $\mathcal{L}^s_{\mathcal{H}\text{-AP}}$ and $\mathcal{L}_{\text{clust.}}$ in HAPPIER performances: we observe a stable increase in $\mathcal{H}$-AP within $0<\lambda<0.5$ 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 $\mathcal{L}_{\text{clust.}}$, and how, when combined, HAPPIER reaches its best performance.

4.4 Qualitative study

We provide here qualitative assessments of HAPPIER, including embedding space analysis and visualization of HAPPIER's retrievals.

t-SNE: organization of the embedding space In Fig. 5, we plot using t-SNE [50,7] 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 show on Fig. 5a the t-SNE visualisation obtained using a baseline method trained on the fine-grained labels, and in Fig. 5b we plot the t-SNE of the embedding space of a model trained with HAPPIER. We cannot observe any clear clusters for the coarse level on Fig. 5a, whereas we can appreciate the the quality of the hierarchical clusters formed on Fig. 5b.

Controlled errors Finally, we showcase in Fig. 6 errors of HAPPIER *vs.* a fine-grained baseline. On Fig. 6a, 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. 6b, we show an example where both models fail 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.



(a) t-SNE visualization of a model trained only on the fine-grained labels.

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

Fig. 5: t-SNE visualisation of the embedding space of two models trained on SOP. Each point is the average embedding of each fine-grained label (object instance) and the colors represent coarse labels (object category, *e.g.* bike, coffee maker).

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

5 Conclusion

In this work, we introduce HAPPIER, a new training method that leverages hierarchical relations between concepts to learn robust rankings. HAPPIER is based on a new metric $\mathcal{H}$-AP that evaluates hierarchical rankings and uses a combination of a smooth upper bound surrogate with theoretical guarantees and a clustering loss to directly optimize it. Extensive experiments show that HAPPIER performs on par to state-of-the-art image retrieval methods on fine-grained metrics and exhibits large improvements *vs.* recent hierarchical methods on hierarchical metrics. Learning more robust rankings reduces the severity of ranking errors, and is qualitatively related to a better organization of the embedding space with HAPPIER. Future works include the adaptation of HAPPIER to the unsupervised setting, *e.g.* for providing a relevant self-training criterion.

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

References

1. Bertinetto, L., Mueller, R., Tertikas, K., Samangooei, S., Lord, N.A.: Making better mistakes: Leveraging class hierarchies with deep networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12506–12515 (2020) 3
2. Brown, A., Xie, W., Kalogeiton, V., Zisserman, A.: Smooth-ap: Smoothing the path towards large-scale image retrieval. In: European Conference on Computer Vision. pp. 677–694. Springer (2020) 1, 3, 6, 8, 9, 10, 11, 12, 28, 29
3. Bruch, S., Zoghi, M., Bendersky, M., Najork, M.: Revisiting approximate metric optimization in the age of deep neural networks. In: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 1241–1244 (2019) 3
4. Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., Hullender, G.: Learning to rank using gradient descent. In: Proceedings of the 22nd International Conference on Machine Learning. p. 89–96. ICML '05, Association for Computing Machinery, New York, NY, USA (2005). https://doi.org/10.1145/1102351.1102363, https://doi.org/10.1145/1102351.1102363 3
5. Burges, C., Ragno, R., Le, Q.: Learning to rank with nonsmooth cost functions. In: Schölkopf, B., Platt, J., Hoffman, T. (eds.) Advances in Neural Information Processing Systems. vol. 19. MIT Press (2006), https://proceedings.neurips.cc/paper/2006/file/af44c4c56f385c43f2529f9b1b018f6a-Paper.pdf 3
6. Cakir, F., He, K., Xia, X., Kulis, B., Sclaroff, S.: Deep metric learning to rank. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1861–1870 (2019) 1, 3, 30
7. Chan, D.M., Rao, R., Huang, F., Canny, J.F.: Gpu accelerated t-distributed stochastic neighbor embedding. Journal of Parallel and Distributed Computing **131**, 1–13 (2019) 13, 31
8. Chang, D., Pang, K., Zheng, Y., Ma, Z., Song, Y.Z., Guo, J.: Your" flamingo" is my" bird": Fine-grained, or not. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11476–11485 (2021) 3, 28
9. Chapelle, O., Chang, Y.: Yahoo! learning to rank challenge overview. In: Proceedings of the learning to rank challenge. pp. 1–24. PMLR (2011) 7
10. Croft, W.B., Metzler, D., Strohman, T.: Search engines: Information retrieval in practice, vol. 520. Addison-Wesley Reading (2010) 3, 31
11. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. pp. 248–255 (2009). https://doi.org/10.1109/CVPR.2009.5206848 28
12. Deng, J., Guo, J., Xue, N., Zafeiriou, S.: Arcface: Additive angular margin loss for deep face recognition. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 4690–4699 (2019) 1, 3
13. Dhall, A., Makarova, A., Ganea, O., Pavllo, D., Greeff, M., Krause, A.: Hierarchical image classification using entailment cone embeddings. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. pp. 836–837 (2020) 3
14. Dupret, G., Piwowarski, B.: A user behavior model for average precision and its generalization to graded judgments. In: Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval. p. 531–538. SIGIR '10, Association for Computing Machinery, New York, NY, USA (2010). https://doi.org/10.1145/1835449.1835538, https://doi.org/10.1145/1835449.1835538 3, 4

15. Dupret, G., Piwowarski, B.: Model based comparison of discounted cumulative gain and average precision. Journal of Discrete Algorithms **18**, 49–62 (2013). https://doi.org/https://doi.org/10.1016/j.jda.2012.10.002, https://www.sciencedirect.com/science/article/pii/S1570866712001372, selected papers from the 18th International Symposium on String Processing and Information Retrieval (SPIRE 2011) 4
16. Fagin, R., Kumar, R., Sivakumar, D.: Comparing top k lists. SIAM Journal on discrete mathematics **17**(1), 134–160 (2003) 30
17. Hadsell, R., Chopra, S., LeCun, Y.: Dimensionality reduction by learning an invariant mapping. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06). vol. 2, pp. 1735–1742. IEEE (2006) 3
18. Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M.H., Brett, M., Haldane, A., del R'ıo, J.F., Wiebe, M., Peterson, P., G'erard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., Oliphant, T.E.: Array programming with NumPy. Nature **585**(7825), 357–362 (Sep 2020). https://doi.org/10.1038/s41586-020-2649-2, https://doi.org/10.1038/s41586-020-2649-2 31
19. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. corr abs/1512.03385 (2015) (2015) 30
20. Hjørland, B.: The foundation of the concept of relevance. Journal of the American Society for Information Science and Technology **61**(2), 217–237 (2010) 3, 5
21. Hunter, J.D.: Matplotlib: A 2d graphics environment. Computing in Science & Engineering **9**(3), 90–95 (2007). https://doi.org/10.1109/MCSE.2007.55 31
22. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of ir techniques. ACM Transactions on Information Systems (TOIS) **20**(4), 422–446 (2002) 3
23. Järvelin, K., Kekäläinen, J.: Ir evaluation methods for retrieving highly relevant documents. In: ACM SIGIR Forum. vol. 51, pp. 243–250. ACM New York, NY, USA (2017) 3
24. Kekäläinen, J., Järvelin, K.: Using graded relevance assessments in ir evaluation. Journal of the American Society for Information Science and Technology **53**(13), 1120–1129 (2002). https://doi.org/https://doi.org/10.1002/asi.10137, https://onlinelibrary.wiley.com/doi/abs/10.1002/asi.10137 3
25. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014) 30
26. Krause, J., Stark, M., Deng, J., Fei-Fei, L.: 3d object representations for fine-grained categorization. In: 4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13). Sydney, Australia (2013) 27
27. Law, M.T., Thome, N., Cord, M.: Learning a distance metric from relative comparisons between quadruplets of images. International Journal of Computer Vision **121**(1), 65–94 (2017) 3
28. Liu, Z., Luo, P., Qiu, S., Wang, X., Tang, X.: Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2016) 27
29. Micikevicius, P., Narang, S., Alben, J., Diamos, G., Elsen, E., Garcia, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G., et al.: Mixed precision training. arXiv preprint arXiv:1710.03740 (2017) 31
30. Miller, G.A.: Wordnet: A lexical database for english. Commun. ACM **38**(11), 39–41 (nov 1995). https://doi.org/10.1145/219717.219748, https://doi.org/10.1145/219717.219748 28

31. Movshovitz-Attias, Y., Toshev, A., Leung, T.K., Ioffe, S., Singh, S.: No fuss distance metric learning using proxies. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 360–368 (2017) 3
32. Musgrave, K., Belongie, S., Lim, S.N.: Pytorch metric learning (2020) 31
33. Oh Song, H., Xiang, Y., Jegelka, S., Savarese, S.: Deep metric learning via lifted structured feature embedding. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4004–4012 (2016) 2, 9, 27, 28
34. P., M.V., Paulus, A., Musil, V., Martius, G., Rolínek, M.: Differentiation of blackbox combinatorial solvers. In: ICLR (2020) 1, 3
35. Parikh, D., Grauman, K.: Relative attributes. In: 2011 International Conference on Computer Vision. pp. 503–510. IEEE (2011) 28
36. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019) 31
37. Qin, T., Liu, T.: Introducing LETOR 4.0 datasets. CoRR **abs/1306.2597** (2013), http://arxiv.org/abs/1306.2597 7
38. Qin, T., Liu, T.Y., Li, H.: A general approximation framework for direct optimization of information retrieval measures. Information Retrieval **13**, 375–397 (2009) 3, 6, 11
39. Radenović, F., Iscen, A., Tolias, G., Avrithis, Y., Chum, O.: Revisiting oxford and paris: Large-scale image retrieval benchmarking. In: CVPR (2018) 28
40. Radenovic, F., Tolias, G., Chum, O.: CNN image retrieval learns from bow: Unsupervised fine-tuning with hard examples. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9905, pp. 3–20. Springer (2016). https://doi.org/10.1007/978-3-319-46448-0_1, https://doi.org/10.1007/978-3-319-46448-0_1 1
41. Radenović, F., Tolias, G., Chum, O.: Cnn image retrieval learns from bow: Unsupervised fine-tuning with hard examples. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) Computer Vision – ECCV 2016. pp. 3–20. Springer International Publishing, Cham (2016) 3
42. Ramzi, E., Thome, N., Rambour, C., Audebert, N., Bitot, X.: Robust and decomposable average precision for image retrieval. Advances in Neural Information Processing Systems **34** (2021) 1, 2, 3, 8, 9, 10, 11, 12, 27, 28, 31
43. Revaud, J., Almazán, J., Rezende, R.S., Souza, C.R.d.: Learning with average precision: Training image retrieval with a listwise loss. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5107–5116 (2019) 1, 3
44. Robertson, S.E., Kanoulas, E., Yilmaz, E.: Extending average precision to graded relevance judgments. In: Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval. pp. 603–610 (2010) 4
45. Rolínek, M., Musil, V., Paulus, A., Vlastelica, M., Michaelis, C., Martius, G.: Optimizing rank-based metrics with blackbox differentiation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7620–7630 (2020) 1, 3, 8
46. Sohn, K.: Improved deep metric learning with multi-class n-pair loss objective. In: Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., Garnett, R. (eds.)

Advances in Neural Information Processing Systems. vol. 29. Curran Associates, Inc. (2016), https://proceedings.neurips.cc/paper/2016/file/6b180037abbebea991d8b1232f8a8ca9-Paper.pdf 1, 3
47. Sun, Y., Zhu, Y., Zhang, Y., Zheng, P., Qiu, X., Zhang, C., Wei, Y.: Dynamic metric learning: Towards a scalable metric space to accommodate multiple semantic scales. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5393–5402 (2021) 2, 4, 9, 10, 11, 12, 28, 29, 30, 31, 32, 33
48. Taylor, M., Guiver, J., Robertson, S., Minka, T.: Softrank: Optimizing non-smooth rank metrics. In: Proceedings of the 2008 International Conference on Web Search and Data Mining. p. 77–86. WSDM '08, Association for Computing Machinery, New York, NY, USA (2008). https://doi.org/10.1145/1341531.1341544, https://doi.org/10.1145/1341531.1341544 3
49. Teh, E.W., DeVries, T., Taylor, G.W.: Proxynca++: Revisiting and revitalizing proxy neighborhood component analysis. In: European Conference on Computer Vision. pp. 448–464. Springer (2020) 1, 2, 3, 8, 9, 10, 11, 12, 31
50. van der Maaten, L., Hinton, G.: Visualizing high-dimensional data using t-sne. Journal of Machine Learning Research **9**, 2579–2605 (2008) 13
51. Van Horn, G., Mac Aodha, O., Song, Y., Cui, Y., Sun, C., Shepard, A., Adam, H., Perona, P., Belongie, S.: The inaturalist species classification and detection dataset. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 8769–8778 (2018) 2, 9, 28
52. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The caltech-ucsd birds-200-2011 dataset. Tech. Rep. CNS-TR-2011-001, California Institute of Technology (2011) 27
53. Wang, H., Wang, Y., Zhou, Z., Ji, X., Gong, D., Zhou, J., Li, Z., Liu, W.: Cosface: Large margin cosine loss for deep face recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 5265–5274 (2018) 1, 3
54. Wang, X., Han, X., Huang, W., Dong, D., Scott, M.R.: Multi-similarity loss with general pair weighting for deep metric learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5022–5030 (2019) 1, 3
55. Wang, X., Zhang, H., Huang, W., Scott, M.R.: Cross-batch memory for embedding learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6388–6397 (2020) 1
56. Wilt, E.W., Harrison, A.V.: Creating a semantic hierarchy of SUN database object labels using WordNet. In: Pham, T., Solomon, L. (eds.) Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications III. vol. 11746, pp. 76 – 83. International Society for Optics and Photonics, SPIE (2021). https://doi.org/10.1117/12.2588827, https://doi.org/10.1117/12.2588827 28
57. Wu, C.Y., Manmatha, R., Smola, A.J., Krahenbuhl, P.: Sampling matters in deep embedding learning. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2840–2848 (2017) 1, 2, 3, 9, 10, 11, 12, 31
58. Yadan, O.: Hydra - a framework for elegantly configuring complex applications. Github (2019), https://github.com/facebookresearch/hydra 31
59. Zhai, A., Wu, H.Y.: Classification is a strong baseline for deep metric learning. arXiv preprint arXiv:1811.12649 (2018) 1, 2, 3, 8, 9, 10, 11, 12, 30, 31

A Method

A.1 $\mathcal{H}$-rank

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

$$\mathcal{H}\text{-rank}(k) = \text{rel}(k) + \sum_{j \in \Omega^+} \min(\text{rel}(k), \text{rel}(j)) \cdot H(s_j - s_k) \ . \tag{7}$$

We detail in Fig. 7 how the $\mathcal{H}$-rank in Eq. (7) is computed in the example from Fig. 2b 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.

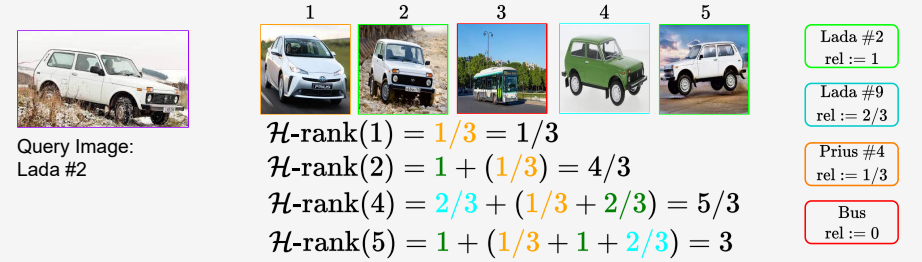


Fig. 7: $\mathcal{H}$-rank for each retrieval results given a "Lada #2" query with relevances of Sec. A.1 and the hierarchical tree of Fig. 2a of the main paper.

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.

A.2 $\mathcal{H}$-AP

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

$$\mathcal{H}\text{-AP} = \frac{1}{\sum_{k \in \Omega^+} \text{rel}(k)} \sum_{k \in \Omega^+} \frac{\mathcal{H}\text{-rank}(k)}{\text{rank}(k)} \tag{8}$$

We illustrate in Fig. 8 how the $\mathcal{H}$-AP is computed for both rankings of Fig. 2b of the main paper. We use the same relevances as in Sec. A.1. 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).

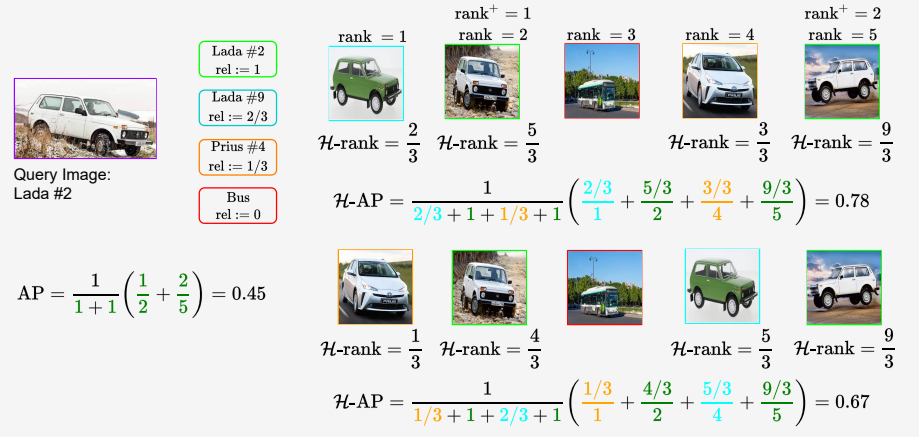


Fig. 8: AP and $\mathcal{H}$-AP for two different rankings when Given a "Lada #2" query and relevances of Sec. A.1. The $\mathcal{H}$-AP of the top row is greater (0.78) than the bottom one's (0.67) as the error in rank$=1$ is less severe for the top row. Whereas the AP is the same for both rankings (0.45).

One property of AP is that it can be interpreted as the area under the precision-recall curve. $\mathcal{H}$-AP from Eq. (8) 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:

$$\mathcal{H}\text{-R@k} = \frac{\sum_{j=1}^{k} \text{rel}(j)}{\sum_{j \in \Omega^+} \text{rel}(j)} \tag{9}$$

$$\mathcal{H}\text{-P@k} = \frac{\sum_{j=1}^{k} \min(\text{rel}(j), \text{rel}(k))}{k \cdot \text{rel}(k)} \tag{10}$$

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

$$\mathcal{H}\text{-AP} = \sum_{k=1}^{|\Omega|} (\mathcal{H}\text{-R@k} - \mathcal{H}\text{-R@k-1}) \times \mathcal{H}\text{-P@k} \tag{11}$$

Eq. (11) recovers Eq. (3) 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.

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. (8).

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

$$\text{AP} = \frac{1}{|\Omega^+|}\sum_{k\in\Omega^+}\frac{\text{rank}^+(k)}{\text{rank}(k)} \tag{12}$$

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

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

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

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

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

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.* $\Omega^{+,l} = \bigcup_{q=l}^{L}\Omega^{(q)}$:

$$\text{AP}^{(l)} = \frac{1}{|\Omega^{+,l}|}\sum_{k\in\Omega^{+,l}}\frac{\text{rank}^{+,l}(k)}{\text{rank}(k)}, \text{rank}^{+,l}(k) = 1 + \sum_{j\in\Omega^{+,l}} H(s_j - s_k) \tag{13}$$

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. (13):

$$\Sigma w\text{AP} = \sum_{l=1}^{L} w_l \cdot \frac{1}{|\Omega^{+,l}|} \sum_{k \in \Omega^{+,l}} \frac{\text{rank}^{+,l}(k)}{\text{rank}(k)} \tag{14}$$

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

$$\Sigma w\text{AP} = \sum_{l=1}^{L} \hat{w}_l \sum_{k \in \Omega^{+,l}} \frac{\text{rank}^{+,l}(k)}{\text{rank}(k)} \tag{15}$$

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 $\text{rank}^{+,l}$.

$$\Sigma w\text{AP} = \sum_{l=1}^{L} \sum_{k \in \Omega^{+}} \frac{\hat{w}_l \cdot \mathbb{1}(k,l) \cdot \text{rank}^{+,l}(k)}{\text{rank}(k)} \tag{16}$$

$$= \sum_{k \in \Omega^{+}} \sum_{l=1}^{L} \frac{\hat{w}_l \cdot \mathbb{1}(k,l) \cdot \text{rank}^{+,l}(k)}{\text{rank}(k)} \tag{17}$$

$$= \sum_{k \in \Omega^{+}} \frac{\sum_{l=1}^{L} \mathbb{1}(k,l) \cdot \hat{w}_l \cdot \text{rank}^{+,l}(k)}{\text{rank}(k)} \tag{18}$$

We replace $\text{rank}^{+,l}$ in Eq. (18) with its definition from Eq. (13):

$$\Sigma w\text{AP} = \sum_{k\in\Omega^+} \frac{\sum_{l=1}^{L} \mathbb{1}(k,l)\cdot\hat{w}_l\cdot\left(1+\sum_{j\in\Omega^{+,l}} H(s_j-s_k)\right)}{\text{rank}(k)} \tag{19}$$

$$= \sum_{k\in\Omega^+} \frac{\sum_{l=1}^{L} \mathbb{1}(k,l)\cdot\hat{w}_l + \sum_{l=1}^{L}\sum_{j\in\Omega^{+,l}} \mathbb{1}(k,l)\cdot\hat{w}_l\cdot H(s_j-s_k)}{\text{rank}(k)} \tag{20}$$

$$= \sum_{k\in\Omega^+} \frac{\sum_{l=1}^{L} \mathbb{1}(k,l)\cdot\hat{w}_l + \sum_{l=1}^{L}\sum_{j\in\Omega^{+}} \mathbb{1}(j,l)\cdot\mathbb{1}(k,l)\cdot\hat{w}_l\cdot H(s_j-s_k)}{\text{rank}(k)} \tag{21}$$

$$= \sum_{k\in\Omega^+} \frac{\sum_{l=1}^{L} \mathbb{1}(k,l)\cdot\hat{w}_l + \sum_{j\in\Omega^{+}}\sum_{l=1}^{L} \mathbb{1}(j,l)\cdot\mathbb{1}(k,l)\cdot\hat{w}_l\cdot H(s_j-s_k)}{\text{rank}(k)} \tag{22}$$

We define the following relevance function:

$$\text{rel}(k) = \sum_{l=1}^{L} \mathbb{1}(k,l)\cdot\hat{w}_l \tag{23}$$

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

$$\sum_{l=1}^{L} \mathbb{1}(j,l)\cdot\mathbb{1}(k,l)\cdot\hat{w}_l = \min(\text{rel}(k),\text{rel}(j)) \tag{24}$$

Using the definition of the relevance function from Eq. (23) and Eq. (24), we can rewrite Eq. (22) with $\mathcal{H}$-rank:

$$\Sigma w\text{AP} = \sum_{k\in\Omega^+} \frac{\text{rel}(k) + \sum_{j\in\Omega^+} \min(\text{rel}(j),\text{rel}(k))\cdot H(s_j-s_k)}{\text{rank}(k)} \tag{25}$$

$$= \sum_{k\in\Omega^+} \frac{\mathcal{H}\text{-rank}(k)}{\text{rank}(k)} \tag{26}$$

Eq. (26) lacks the normalization constant $\sum_{k\in\Omega^+}\text{rel}(k)$ in order to have the same shape as $\mathcal{H}$-AP in Eq. (8). So we must prove that $\sum_{k\in\Omega^+}\text{rel}(k)=1$:

$$\sum_{k\in\Omega^+}\text{rel}(k)=\sum_{k\in\Omega^+}\sum_{l=1}^{L}\mathbb{1}(k,l)\cdot\hat{w}_l \tag{27}$$

$$=\sum_{l=1}^{L}|\Omega^{(l)}|\sum_{p=1}^{l}\hat{w}_p \tag{28}$$

$$=\sum_{l=1}^{L}|\Omega^{(l)}|\sum_{p=1}^{l}\frac{w_p}{|\Omega^{+,p}|} \tag{29}$$

$$=\sum_{l=1}^{L}|\Omega^{(l)}|\sum_{p=1}^{l}\frac{w_p}{|\bigcup_{q=p}^{L}\Omega^{(q)}|} \tag{30}$$

$$=\sum_{l=1}^{L}|\Omega^{(l)}|\sum_{p=1}^{l}\frac{w_p}{\sum_{q=p}^{L}|\Omega^{(q)}|} \tag{31}$$

$$=\sum_{l=1}^{L}\sum_{p=1}^{l}\frac{|\Omega^{(l)}|\cdot w_p}{\sum_{q=p}^{L}|\Omega^{(q)}|} \tag{32}$$

$$=\sum_{p=1}^{L}\sum_{l=p}^{L}\frac{|\Omega^{(l)}|\cdot w_p}{\sum_{q=p}^{L}|\Omega^{(q)}|} \tag{33}$$

$$=\sum_{p=1}^{L}w_p\cdot\frac{\sum_{l=p}^{L}|\Omega^{(l)}|}{\sum_{q=p}^{L}|\Omega^{(q)}|} \tag{34}$$

$$=\sum_{p=1}^{L}w_p=1 \tag{35}$$

We have proved that $\Sigma w\text{AP}=\mathcal{H}$-AP with the relevance function of Eq. (23):

$$\Sigma w\text{AP}=\frac{1}{\sum_{k\in\Omega^+}\text{rel}(k)}\sum_{k\in\Omega^+}\frac{\mathcal{H}\text{-rank}(k)}{\text{rank}(k)}=\mathcal{H}\text{-AP} \tag{36}$$

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

$$\text{rel}(k)=\sum_{p=1}^{L}\mathbb{1}(k,p)\cdot\hat{w}_p=\sum_{p=1}^{l}\cdot\hat{w}_p=\sum_{p=1}^{l}\frac{w_p}{|\Omega^{+,p}|} \tag{37}$$

i.e. the relevance of Eq. (23) is the same as the relevance of Property 1. This concludes the proof of Property 1. □

A.3 Direct optimisation of $\mathcal{H}$-AP

Decomposing $\mathcal{H}$-rank and rank We have $\Omega^+ = \bigcup_{q=1}^{L} \Omega^{(q)}$, for an instance $k \in \Omega^{(l)}$ we can define the following subsets: $\Omega^> = \bigcup_{q=l+1}^{L} \Omega^{(q)}$ and $\Omega^\leq = \bigcup_{q=1}^{l} \Omega^{(q)}$, so that $\Omega^+ = \Omega^> \cup \Omega^\leq$. So we can rewrite $\mathcal{H}$-rank:

$$\begin{aligned}\mathcal{H}\text{-rank}(k) &= \text{rel}(k) + \sum_{j \in \Omega^+} \min(\text{rel}(k), \text{rel}(j)) \cdot H(s_j - s_k) \\ &= \underbrace{\sum_{j \in \Omega^>} \min(\text{rel}(k), \text{rel}(j)) \cdot H(s_j - s_k)}_{\mathcal{H}\text{-rank}^>} \\ &\quad \underbrace{+\text{rel}(k) + \sum_{j \in \Omega^\leq} \min(\text{rel}(k), \text{rel}(j)) \cdot H(s_j - s_k)}_{\mathcal{H}\text{-rank}^\leq}\end{aligned}$$

Similarly we can define $\Omega^\geq = \bigcup_{q=l}^{L} \Omega^{(q)}$ and $\Omega^< = \bigcup_{q=0}^{l-1} \Omega^{(q)}$, with $\Omega^+ = \Omega^\geq \cup \Omega^<$. So we can rewrite rank:

$$\begin{aligned}\text{rank}(k) &= 1 + \sum_{k \in \Omega} H(s_j - s_k) \\ &= \underbrace{1 + \sum_{k \in \Omega^\geq} H(s_j - s_k)}_{\text{rank}^\geq} \underbrace{+ \sum_{k \in \Omega^<} H(s_j - s_k)}_{\text{rank}^<}\end{aligned}$$

Gradients for $\mathcal{L}_{\mathcal{H}\text{-AP}}$ We further decompose $\mathcal{L}_{\mathcal{H}\text{-AP}}$ from Eq. 5 of the main paper, using $\mathcal{H}\text{-rank}^\leq(k) = \mathcal{H}\text{-rank}^=(k) + \mathcal{H}\text{-rank}^<(k)$, $\text{rank}^\geq(k) = \text{rank}^>(k) + \text{rank}^=(k)$:

$$\mathcal{L}_{\mathcal{H}\text{-AP}} = 1 - \frac{1}{\sum_{k \in \Omega^+} \text{rel}(k)} \sum_{k \in \Omega^+} \frac{\mathcal{H}\text{-rank}^>(k) + \mathcal{H}\text{-rank}^=(k) + \mathcal{H}\text{-rank}^<(k)}{\text{rank}^>(k) + \text{rank}^=(k) + \text{rank}^<(k) + \text{rank}^-(k)}$$

Table 7: Decomposition of $\mathcal{H}$-AP for optimization.

	$\mathcal{H}\text{-rank}^>$	$\text{rank}^<$	rank^-	$\mathcal{H}\text{-rank}^=$	$\mathcal{H}\text{-rank}^<$	$\text{rank}^>$	$\text{rank}^=$
Optimization	✓	✓	✓	✗	✗	✗	✗

We choose to only optimize with respect to the terms indicated with ✓ in Tab. 7.

rank$^-$(k): $\frac{\partial \mathcal{L}_{\mathcal{H}\text{-AP}}}{\partial \text{rank}^-(k)} \propto \frac{\mathcal{H}\text{-rank}(k)}{\text{rank}(k)^2} > 0$ which means that in order to decrease $\mathcal{L}_{\mathcal{H}\text{-AP}}$ we must lower rank$^-$, which is an expected behaviour, as it will force k to have a better ranking if it ranked after negative instances (in Ω^-).

rank$^<$(k): if we suppose that $\mathcal{H}$-rank$^<$ is a constant, then $\frac{\partial \mathcal{L}_{\mathcal{H}\text{-AP}}}{\partial \text{rank}^<(k)} \propto \frac{\mathcal{H}\text{-rank}(k)}{\text{rank}(k)^2} > 0$ which means that in order to decrease $\mathcal{L}_{\mathcal{H}\text{-AP}}$ we must lower rank$^<$, which is an expected behaviour, as it will force k to have a better ranking if it ranked after negative instances (in $\Omega^<$).

$\mathcal{H}$-rank$^>$(k): if we suppose that rank$^>$ is a constant, $\frac{\partial \mathcal{L}_{\mathcal{H}\text{-AP}}}{\partial \mathcal{H}\text{-rank}^>(k)} \propto \frac{-1}{\text{rank}(k)} < 0$ which means that in order to decrease $\mathcal{L}_{\mathcal{H}\text{-AP}}$ we must increase $\mathcal{H}$-rank$^>$, which is an expected behaviour, as it will force k to be ranked after other instances of higher relevance (in $\Omega^>$).

We choose to not optimize with respect to $\mathcal{H}$-rank$^=$, $\mathcal{H}$-rank$^<$, rank$^>$, rank$^=$.

rank$^=$ & $\mathcal{H}$-rank$^=$: Optimizing through rank$^=$ has no impact so we choose not to optimize it, indeed $\frac{\partial \mathcal{L}_{\mathcal{H}\text{-AP}}}{\partial \text{rank}^=} = 0$. This is the case because inversions between instances of same relevance has no impact on $\mathcal{H}$-AP. This is also the case for $\mathcal{H}$-rank$^=$.

$\mathcal{H}$-rank$^<$(k): $\mathcal{H}$-rank$^<(k)$ depends on rank$^<(k)$ and the relevance of the other instances that are before. We note that $0 < \frac{\partial \mathcal{H}\text{-rank}^<(k)}{\partial \text{rank}(k)} < \text{rel}(k)$ indeed when the rank$^<$ increases $\mathcal{H}$-rank$^<$ increases and the increase rate can not be equal or greater than rel(k)

$$\frac{\partial \mathcal{L}_{\mathcal{H}\text{-AP}}}{\partial \text{rank}^<(k)} \propto -\Bigg(\overbrace{\left(\frac{\partial \mathcal{H}\text{-rank}^<(k)}{\partial \text{rank}^<(k)} - \text{rel}(k) \right) \cdot \text{rank}^>(k)}^{a} \tag{38}$$

$$+ \overbrace{\left(\frac{\partial \mathcal{H}\text{-rank}^<(k)}{\partial \text{rank}^<(k)} - \text{rel}(k) \right) \cdot \text{rank}^=(k)}^{b} \tag{39}$$

$$+ \overbrace{\left(\frac{\partial \mathcal{H}\text{-rank}^<(k)}{\partial \text{rank}^<(k)} \cdot \text{rank}^<(k) - \mathcal{H}\text{-rank}^<(k) \right)}^{c} \tag{40}$$

$$+ \overbrace{\frac{\partial \mathcal{H}\text{-rank}^<(k)}{\partial \text{rank}^<(k)} \cdot \text{rank}^-(k)}^{d} \Bigg) / \text{rank}(k)^2 \tag{41}$$

When optimizing through $\mathcal{H}$-rank$^<$ we can no longer explicitly control the sign of $\frac{\partial \mathcal{L}_{\mathcal{H}\text{-AP}}}{\partial \text{rank}^<(k)}$. For example if a and b are null (*i.e.* not instances of higher or equal relevance are above k), d remains and is greater than 0 and c can be greater than 0 resulting in an overall negative gradient, which is an unexpected behaviour. This is why we choose to not optimize through $\mathcal{H}$-rank$^<$.

rank$^>(k)$: We have $\mathcal{H}$-rank$^>(k) = \text{rel}(k) \cdot \text{rank}^>(k)$ indeed all instances j ranked before k have a strictly higher relevance, *i.e.* $\min(\text{rel}(j),\text{rel}(k)) = \text{rel}(k)$, so we can write:

$$\frac{\partial \mathcal{L}_{\mathcal{H}\text{-AP}}}{\partial \text{rank}^>(k)} \propto \frac{\overbrace{\mathcal{H}\text{-rank}^<(k) - \text{rel}(k) \cdot \text{rank}^<(k)}^{<0} - \text{rel}(k) \cdot \text{rank}^-(k)}{\text{rank}(k)^2} < 0 \tag{42}$$

Optimizing trough rank$^>$ instead of only $\mathcal{H}$-rank$^>$ diminishes the magnitude of the resulting gradient, so we decide to not optimize through rank$^>$.

Approximating $\mathcal{H}$-rank$^>$ In order to have a lower bound on $\mathcal{H}$-rank$^>$ we approximate the Heaviside step function H with a smooth lower bound:

$$H_s^>(t) = \begin{cases} \gamma \cdot t, & \text{if } t < 0 \\ \max(\nu \cdot t + \mu, 1), & \text{if } t \geq 0 \end{cases} \tag{43}$$

$H_s^>$ is illustrated in Fig. 9a. Using $H_s^>$ we can approximate $\mathcal{H}$-rank$^>$: $\mathcal{H}\text{-rank}_s^>(k) = \text{rel}(k) + \sum_{j \in \Omega^+} \min(\text{rel}(j),\text{rel}(k)) H_s^>(s_j - s_k)$. Because $H_s^>(t) \leq H(t)$: $\mathcal{H}\text{-rank}_s^>(k) \leq \mathcal{H}\text{-rank}^>$. In our experiments we use: $\gamma = 10$, $\nu = 25$, $\mu = 0.5$.

Approximating rank$^<$ In order to have an upper bound on rank$^<$ we approximate the Heaviside with a smooth upper bound as given in [42]:

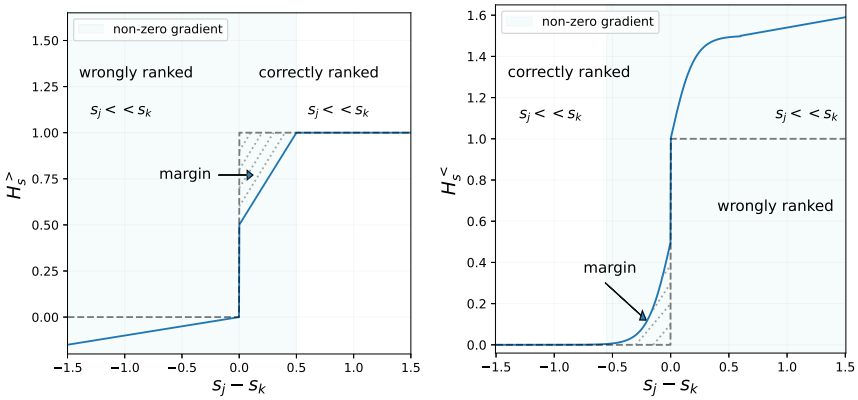
$$H_s^<(t) = \begin{cases} \sigma(\frac{t}{\tau}) & \text{if } t \leq 0, \quad \text{where } \sigma \text{ is the sigmoid function} \\ \sigma(\frac{t}{\tau}) + 0.5 & \text{if } t \in [0;\delta] \quad \text{with } \delta \geq 0 \\ \rho \cdot (t - \delta) + \sigma(\frac{\delta}{\tau}) + 0.5 & \text{if } t > \delta \end{cases} \tag{44}$$

$H_s^<$ is illustrated in Fig. 9a. Using $H_s^<$ we can approximate rank$^<$: $\text{rank}_s^<(k) = 1 + \sum_{j \in \Omega} H_s^<(s_j - s_k)$. Because $H_s^<(t) \geq H(t)$: $\text{rank}_s^<(k) \geq \text{rank}^<$. We use the hyper-parameters: $\tau = 0.01$, $\rho = 100$, $\delta = 0.05$.

We illustrate in Fig. 9a $H_s^>$ and in Fig. 9a $H_s^<$ *vs.* $s_j - s_k$. The margins denote the fact the even when the instance k is correctly ranked (lower cosine similarity than j in Fig. 9a and higher in Fig. 9a) we still want to backbropagate gradient which leads to more robust training.

A.4 Discussion

HAPPIER requires the definition of the relevance function. In our work, we leverage the hierarchical tree between concepts to this end. Is this a strong assumption? We argue that the access to a hierarchical tree is not a prohibitive factor. Hierarchical trees are available for a surprising number of datasets (CUB-200-2011 [52], Cars196 [26], InShop [28], SOP [33]), including *large scale* ones



(a) Illustration of $H_s^>$ in Eq. (43). (b) Illustration of $H_s^<$ in Eq. (43).

Fig. 9: Illustrations of the two approximations of the Heaviside step function used to approximate $\mathcal{H}$-rank$^>$ and rank$^<$.

(iNaturalist [51], the three DyML datasets [47] and also Imagenet [11]). Even when hierarchical relations are not directly available, they are not that difficult to obtain since the tree complexity depends only on the number of classes and not of examples. Hierarchical relations can be semi-automatically obtained by grouping fine-grained labels in existing datasets, as was previously done by *e.g.* [8]. For instance, while hierarchical labels are not directly available in scene or landmarks datasets [39], this could be extended to them at a reasonable cost, *e.g.* in Paris6k "Sacre Coeur" might be considered closer to "Notre Dame" than to the "Moulin Rouge". The large lexical database Wordnet [30] can also be used to define hierarchies between labels and define semantic similarity, as in Imagenet [11] or the SUN database [56]. Furthermore, our approach can be extended to leverage general knowledge beyond hierarchical taxonomies, by defining more general relevance functions built on *e.g.* continuous similarities or attributes [35].

B Experiments

B.1 Datasets

Stanford Online Product (SOP) [33] is a standard dataset for Image Retrieval it has two levels of semantic scales, the object Id (fine) and the object category (coarse). It depicts Ebay online objects, with 120053 images of 22634 objects (Id) classified into 12 (coarse) categories (*e.g.* bikes, coffee makers *etc.*), see Fig. 10. We use the reference train and test splits from [33]. The dataset can be downloaded at: `https://cvgl.stanford.edu/projects/lifted_struct/`.

iNaturalist-2018 Base/Full iNaturalist-2018 is a dataset that has been used for Image Retrieval in recent works [2,42]. It depicts animals, plants, mushroom *etc.*

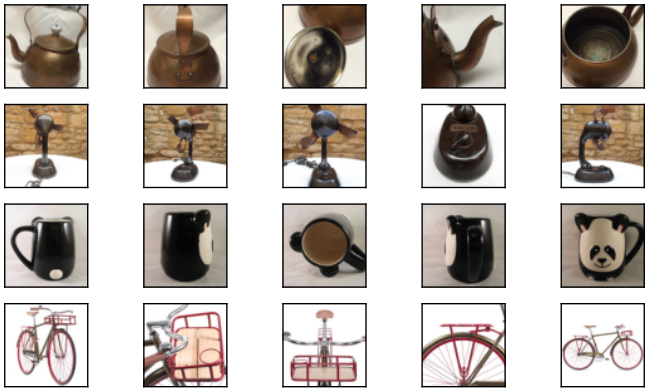


Fig. 10: Images from Stanford Online Products.

in wildlife, see Fig. 11, it has in total 461939 images and 8142 fine-grained classes ("Species"). We use two different sets of annotations: a set of annotations with 2 semantic levels the species (fine) and intermediate scale (coarse), we term this dataset iNat-base, and the full biological taxonomy which consists of 7 semantic levels ("Species", "Genus" ...) we term this dataset iNat-full. We use the standard Image Retrieval splits from [2]. The dataset can be downloaded at: github.com/visipedia/inat_comp, and the retrieval splits at: drive.google.com.

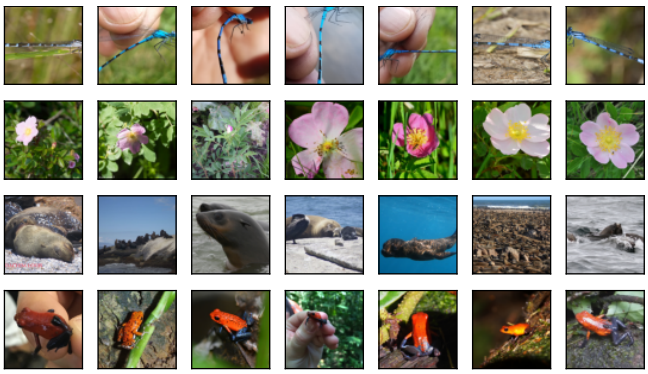


Fig. 11: Images from iNaturalist-2018.

DyML-datasets The DyML benchmark [47] is composed of three datasets, DyML-V that depicts vehicles, DyML-A that depicts animals, DyML-P that depicts online products. The training set has three levels of semantic ($L = 3$), and each image is annotated with the label corresponding to each level (like SOP and iNat-base/full), however the test protocol is different. At test time for each dataset there is three sub-datasets, each sub-dataset aims at evaluating the

model on a specific hierarchical level (*e.g.* "Fine"), so we can only compute binary metrics on each sub-dataset. We describe in Tab. 8 the statistics of the train and test datasets. The three datasets can be downloaded at: `onedrive.live.com`.

Table 8: Statistics of the three train and test DyML benchmarks [47].

Datasets		DyML-Vehicle		DyML-Animal		DyML-Product	
		train	test	train	test	train	test
Coarse	Classes	5	6	5	5	36	6
	Images	343.1 K	5.9 K	407.8 K	12.5 K	747.1 K	1.5 K
Middle	Classes	89	127	28	17	169	37
	Images	343.1 K	34.3 K	407.8 K	23.1 K	747.1 K	1.5 K
Fine	Classes	36,301	8,183	495	162	1,609	315
	Images	343.1 K	63.5 K	407.8 K	11.3 K	747.1 K	1.5 K

B.2 Implementation details

SOP & iNat-base/full Our model is a ResNet-50 [19] pretrained on Imagenet to which we append a `LayerNormalization` layer with no affine parameters after the (average) pooling and a Linear layer that reduces the embeddings size from 2048 to 512. We use the Adam [25] optimizer with a base learning rate of $1e^{-5}$ and weight decay of $1e^{-4}$ for SOP and a base learning rate of $1e^{-5}$ and weight decay of $4e^{-4}$ for iNat-base/full. The learning rate is decreased using cosine annealing decay, for 75 epochs on SOP and 100 epochs on iNat-base/full. We "warm up" our model for 5 epochs, *i.e.* the pretrained weights are not optimized. We use standard data augmentation: `RandomResizedCrop` and `RandomHorizontalFlip`, with a final crop size of 224, at test time we use `CenterCrop`. We set the random seed to 0 in all our experiments. We use a fixed batch size of 256 and use the hard sampling strategy from [6] on SOP and the standard class balanced sampling [59] (4 instances per class) on iNat-base/full.

DyML We use a ResNet-34 [19] randomly initialized on DyML-V&A and pretrained on Imagenet for DyML-P, following [47]. We use an SGD optimizer with Nesterov momentum (0.9), a base learning rate of 0.1 on DyML-V&A and 0.01 on DyML-P with a weight decay of $1e^{-4}$. We use cosine annealing decay to reduce the learning rate for 100 epochs on DyML-V&A and 20 on DyML-P. We use the same data augmentation and random seed as for SOP and iNat-base. We also use the class balanced sampling (4 instances per class) with a fixed batch size of 256.

B.3 Metrics

The ASI [16] 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:

$$SI(n) = \frac{|\{a_1,...,a_n\} \cap \{b_1,...,b_n\}|}{n}$$
$$ASI = \frac{1}{N}\sum_{n=1}^{N} SI(n)$$

The NDCG [10] is the reference metric in information retrieval, we define it using the semantic level l of each instance:

$$DCG = \sum_{k \in \Omega^+} \frac{2^l - 1}{\log_2(1+\text{rank}(k))}, \text{with } k \in \Omega^{(l)}.$$
$$NDCG = \frac{DCG}{\max_{\text{ranking}} DCG}$$

To compute the AP for the semantic level l we consider that all instances with semantic levels $\geq l$ are positives:

$$AP^{(l)} = \sum_{k \in \bigcup_{q=l}^{L} \Omega^{(q)}} \frac{\text{rank}^l(k)}{\text{rank}(k)}, \text{where } \text{rank}^l(k) = 1 + \sum_{j \in \bigcup_{q=l}^{L} \Omega^{(q)}} H(s_j - s_k)$$

B.4 Source Code

Our code is based on `PyTorch` [36]. We use utilities from `Pytorch Metric Learning` [32] *e.g.* for samplers and losses, `Hydra` [58] to handle configuration files (Yaml), `tsnecuda` [7] to compute t-SNE reductions using GPUs and standard Python libraries such as `NumPy` [18] or `Matplotlib` [21].

We use the publicly available implementations of the NSM loss [59][3] which is under an Apache-2.0 license, of NCA++[49][4] which is under an MIT license, of ROADMAP [42][5] which is under an MIT license, we use the implementation of `Pytorch Metric Learning` [32][6] for the TL_{SH} [57] (MIT license), and finally we have implemented the CSL [47] after discussion with the authors and we will make it part of our repository.

We had access to both Nvidia Quadro RTX 5000 and Tesla V-100 (16 GiB GPUs). We use mixed precision training [29], which is native to `PyTorch`, to accelerate training, making our models train for up to 7 hours on Stanford Online Products, 25 hours on iNaturalist-2018, less than 20 hours on both DyML-A and DyML-V and 6 hours on DyML-P.

[3] https://github.com/azgo14/classification_metric_learning
[4] https://github.com/euwern/proxynca_pp
[5] https://github.com/elias-ramzi/ROADMAP
[6] https://github.com/KevinMusgrave/pytorch-metric-learning

B.5 On DyML results

Their is no public code available to reproduce the results of [47]. After personal correspondence with the authors, we have been able to re-implement the CSL method from [47]. We report the differences in performances between our results and theirs in Tab. 9. Our implementation of CSL performs better on the three datasets which is the results of our better training recipes detailed in Sec. B.2. Our discussions with the authors of [47] confirmed that the performances obtained with our re-implementation of CSL are valid and representative of the method's potential.

Table 9: Difference in performances for CSL between results reported in [47] and our experiments on the DyML benchmarks.

Method	DyML-Vehicle			DyML-Animal			DyML-Product		
	mAP	ASI	R@1	mAP	ASI	R@1	mAP	ASI	R@1
CSL [47]	12.1	23.0	25.2	31.0	45.2	52.3	28.7	29.0	54.3
CSL (ours)	30.0	43.6	87.1	40.8	46.3	60.9	31.1	40.7	52.7

C Qualitative results

C.1 Robustness to λ

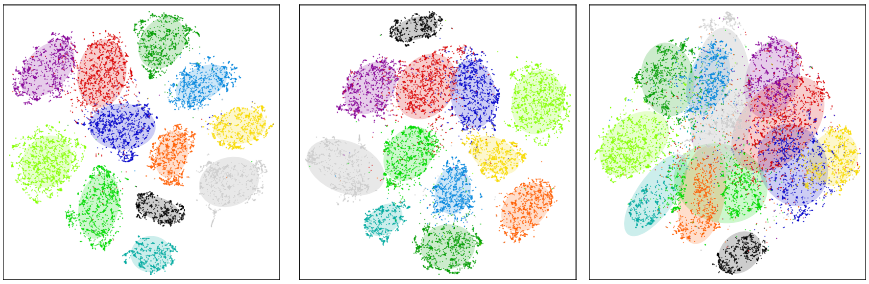
Fig. 4b of the main paper illustrates that HAPPIER is robust with respect to λ with performances increasing for most values between 0.1 and 0.9. In addition, we also show in Fig. 12 that for $0 < \lambda < 0.9$ HAPPIER leads to a better organization of the embedding space than a fine-grained baseline (see Fig. 4a in main paper). This is expected since the lower λ is, the more emphasis is put on optimizing $\mathcal{H}$-AP, which organizes the embedding space in a hierarchical structure.

C.2 Comparison of HAPPIER *vs.* CSL

In Fig. 13, we compare HAPPIER against the hierarchical image retrieval method CSL [47]. We observe qualitative improvements where HAPPIER results in a better ranking. This highlights the benefit of optimizing directly a hierarchical metric, *i.e.* $\mathcal{H}$-AP, rather than optimizing a proxy based triplet loss as in CSL.

C.3 Controlled errors: iNat-base

We showcase in Fig. 14 errors of HAPPIER *vs.* a fine-grained baseline on iNat-base. On Fig. 14a, 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. 14b, we show an example where both models fail to retrieve the correct fine-grained instances, however the model trained with HAPPIER retrieves images of bikes that are semantically more similar to the query.



(a) $\lambda=0.3$ (b) $\lambda=0.5$ (c) $\lambda=0.9$

Fig. 12: t-SNE visualisation of the embedding space of models trained with HAPPIER on SOP with different values of λ. Each point is the average embedding of each fine-grained label (object instance) and the colors represent coarse labels (object category, *e.g.* bike, coffee maker).

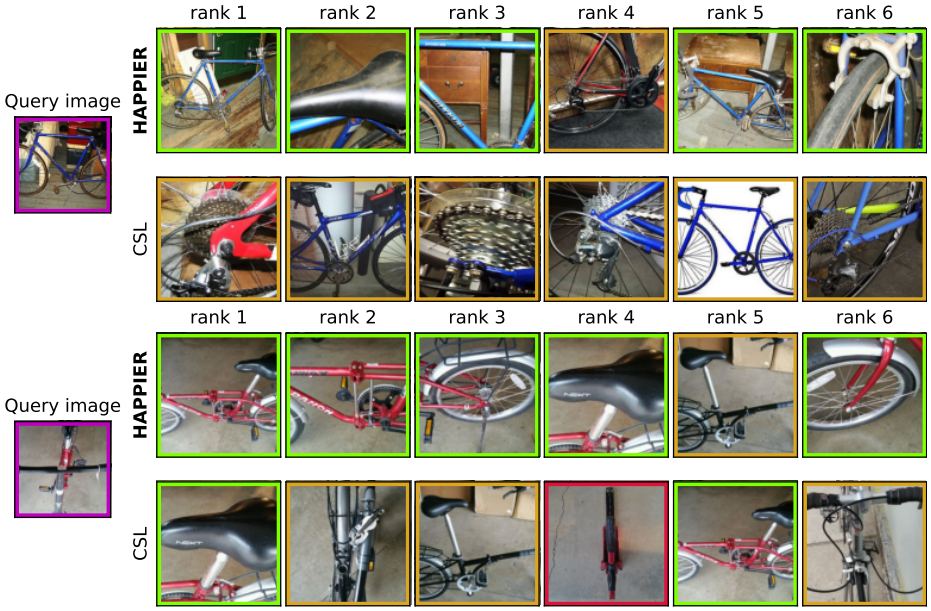
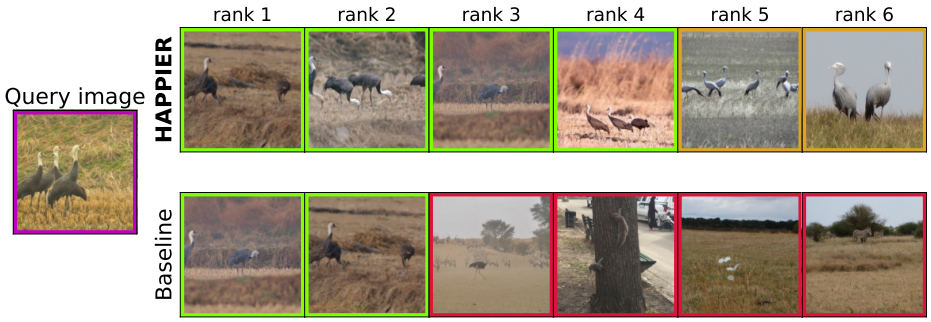


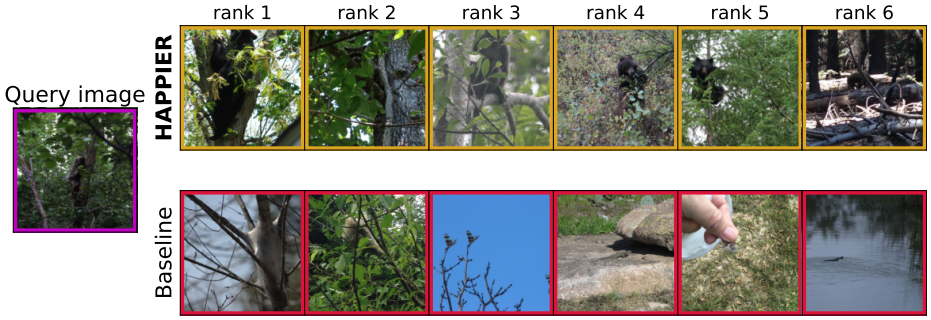
Fig. 13: Qualitative comparison of HAPPIER *vs.* CSL [47] on SOP

C.4 Controlled errors: iNat-full

We illustrate in Figs. 15 and 16 an example of a query image and the top 25 retrieved results on iNat-full ($L=7$). Given the same query both models failed to retrieve the correct fine-grained images (that would be in $\Omega^{(7)}$). The standard model in Fig. 16 retrieves images that are semantically more distant than the images retrieved with HAPPIER in Fig. 15. For example HAPPIER retrieves



(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 (in orange).



(b) In this example, the models fail to retrieve the correct fine grained images. However HAPPIER still retrieves images with the same coarse label (in orange) whereas the baseline retrieves images that are dissimilar semantically to the query (in red).

Fig. 14: Qualitative examples of failure cases from a standard fine-grained model corrected by training with HAPPIER.

images that are either in $\Omega^{(5)}$ or $\Omega^{(4)}$ (only one instance is in $\Omega^{(3)}$) whereas the standard model retrieves instances that are in $\Omega^{(2)}$ or $\Omega^{(1)}$.

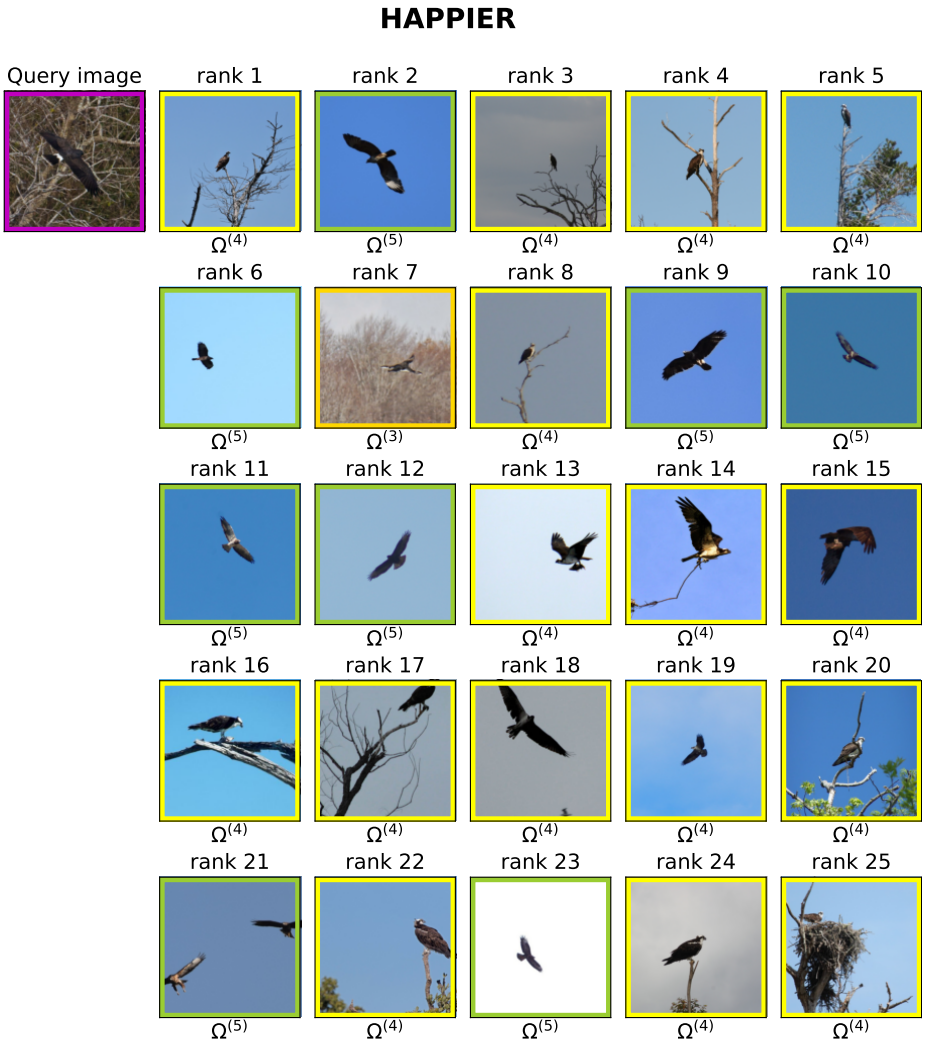


Fig. 15: Images retrieved for the query image by a model trained with **HAPPIER** on iNat-full ($L=7$).

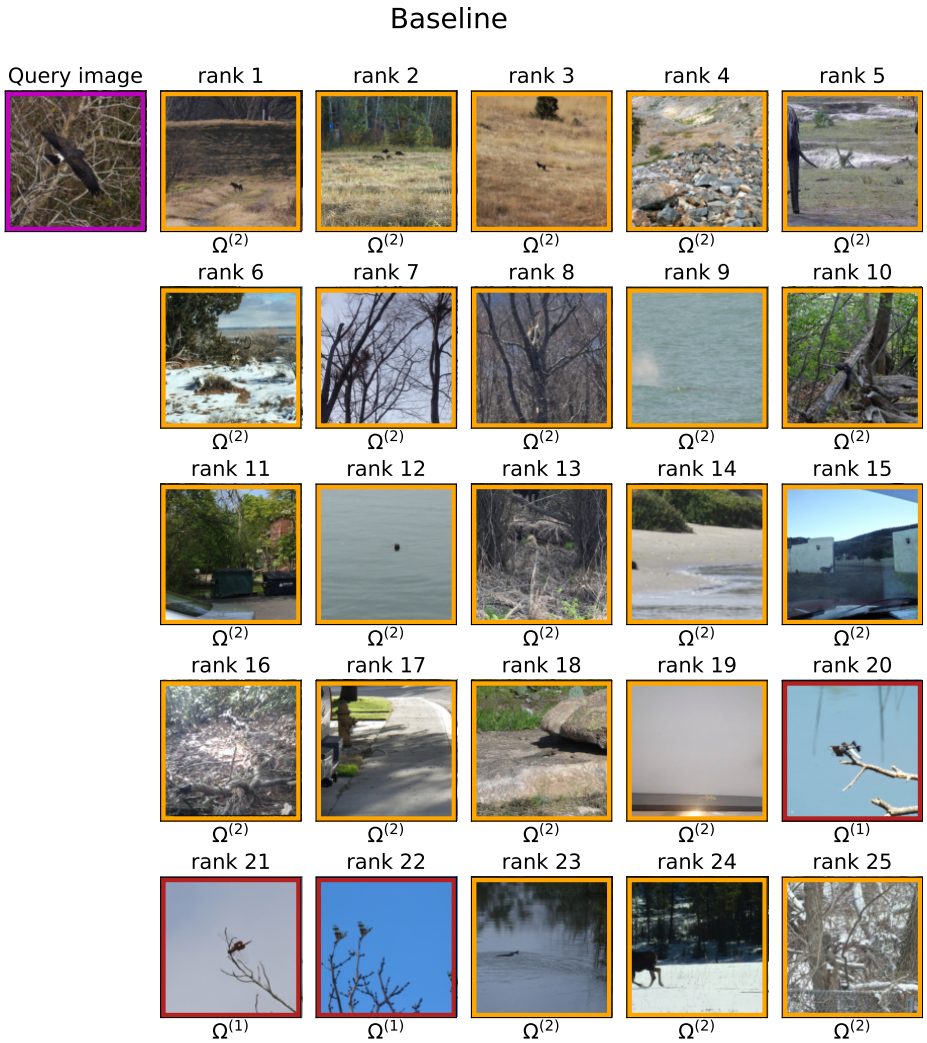


Fig. 16: Images retrieved for the query image by a model trained with standard model on iNat-full ($L=7$).