Hashing as Tie-Aware Learning to Rank

Kun He  Fatih Cakir  Sarah Adel Bargal  Stan Sclaroff
Department of Computer Science, Boston University
{hekun, fcakir, sbargal, sclaroff}@cs.bu.edu

Abstract

Hashing, or learning binary embeddings of data, is frequently used in nearest neighbor retrieval. In this paper, we develop learning to rank formulations for hashing, aimed at directly optimizing ranking-based evaluation metrics such as Average Precision (AP) and Normalized Discounted Cumulative Gain (NDCG). We first observe that the integer-valued Hamming distance often leads to tied rankings, and propose to use tie-aware versions of AP and NDCG to evaluate hashing for retrieval. Then, to optimize tie-aware ranking metrics, we derive their continuous relaxations, and perform gradient-based optimization with deep neural networks. Our results establish the new state-of-the-art for image retrieval by Hamming ranking in common benchmarks.

1. Introduction

In this paper, we consider the problem of hashing, which is concerned with learning binary embeddings of data in order to enable fast approximate nearest neighbor retrieval. We take a task-driven approach, and seek to optimize learning objectives that closely match test-time performance measures. Nearest neighbor retrieval performance is frequently measured using ranking-based evaluation metrics, such as Average Precision (AP) and Normalized Discounted Cumulative Gain (NDCG). We first observe that the integer-valued Hamming distance often leads to tied rankings, and propose to use tie-aware versions of AP and NDCG to evaluate hashing for retrieval. Then, to optimize tie-aware ranking metrics, we derive their continuous relaxations, and perform gradient-based optimization with deep neural networks. Our results establish the new state-of-the-art for image retrieval by Hamming ranking in common benchmarks.

Figure 1: When applying hashing for nearest neighbor retrieval, the integer-valued Hamming distance produces ties (items that share the same distance). If left uncontrolled, different tie-breaking strategies could give drastically different values of the evaluation metric, e.g., AP. We address this issue by using tie-aware ranking metrics that implicitly average over all the permutations in closed form. We further use tie-aware ranking metrics as optimization objectives in deep hashing networks, leading to state-of-the-art results.

Our natural next step is to learn hash functions by optimizing tie-aware ranking metrics. This can be seen as an instance of learning to rank with listwise loss functions, which is advantageous compared to many other ranking-inspired hashing formulations. To solve the associated discrete and NP-hard optimization problems, we relax the problems into their continuous counterparts where closed-form gradients are available, and then perform gradient-based optimization with deep neural networks. We specifically study the optimization of AP and NDCG, two ranking metrics that are naturally tied.
widely used in evaluating nearest neighbor retrieval performance. Our results establish the new state-of-the-art for these metrics in common image retrieval benchmarks.

2. Related Work

Hashing is a widely used approach for practical nearest neighbor retrieval [39], thanks to the efficiency of evaluating Hamming distances using bitwise operations, as well as the low memory and storage footprint. It has been theoretically demonstrated [1] that data-dependent hashing methods outperform data-independent ones such as Locality Sensitive Hashing [14]. We tackle the supervised hashing problem, also known as affinity-based hashing [18, 25, 30], where supervision is given in the form of pairwise affinities. Regarding optimization, the discrete nature of hashing usually results in NP-hard problems. Our solution uses continuous relaxations, which is in line with relaxation-based methods, e.g. [4, 18, 25], but differs from alternating methods that preserve the discrete constraints [22, 29, 30] and two-step methods [6, 23, 48].

Supervised hashing can be cast as a distance metric learning problem [29], which itself can be formulated as learning to rank [21, 27]. Optimizing ranking metrics such as AP and NDCG has received much attention in the learning to rank literature. For instance, surrogates of AP and NDCG can be optimized in the structural SVM framework [10, 45], and bound optimization algorithms exist for NDCG [38]. Alternatively, there are gradient-based methods based on smoothing or approximating these metrics [2, 11, 19, 35]. Recently, [36] tackles few-shot classification by optimizing AP using the direct loss minimization framework [34]. These methods did not consider applications in hashing.

In the learning to hash literature, different strategies have been proposed to handle the difficulties in optimizing listwise ranking metrics. For example, [40] decomposes listwise supervision into local triplets, [22, 44] use structural SVMs to optimize surrogate losses, [33] maximizes precision at the top, and [41, 47] optimize NDCG surrogates. In other recent methods using deep neural networks, the learning objectives are not designed to match ranking evaluation metrics, e.g. [4, 20, 42, 48]. In contrast, we directly optimize listwise ranking metrics using deep neural networks.

Key to our formulation is the observation that the integer-valued Hamming distance results in rankings with ties. However, this fact is not widely taken into consideration in previous work. Ties can be sidestepped by using weighted Hamming distance [22, 46], but at the cost of reduced efficiency. Fortunately, tie-aware versions of common ranking metrics have been found in the information retrieval literature [28]. Inspired by such results, we propose to optimize tie-aware ranking metrics on Hamming distances. Our gradient-based optimization uses a recent differentiable histogram binning technique [4, 5, 37].

3. Hashing as Tie-Aware Ranking

3.1. Preliminaries

Learning to hash. In learning to hash, we wish to learn a hash mapping \( \Phi : X \to H^{b} \), where \( X \) is the feature space, and \( H^{b} = \{-1, 1\}^{b} \) is the b-dimensional Hamming space. A hash mapping \( \Phi \) induces the Hamming distance \( d_{\Phi} : X \times X \to \{0, 1, \ldots, b\} \) as

\[
d_{\Phi}(x, x') = \frac{1}{2} \langle b - \Phi(x)^{\top} \Phi(x') \rangle.
\]

We consider a supervised learning setting, or supervised hashing, where supervision is specified using pairwise affinities. Formally, we assume access to an affinity oracle \( \mathcal{A} \), whose value indicates a notion of similarity: two examples \( x_i, x_j \in X \) are called similar if \( \mathcal{A}(x_i, x_j) > 0 \), and dissimilar when \( \mathcal{A}(x_i, x_j) = 0 \). In this paper, we restrict \( \mathcal{A} \) to take values from a finite set \( \mathcal{V} \), which covers two important special cases. First, \( \mathcal{V} = \{0, 1\} \), or binary affinities, are extensively studied in the current literature. Binary affinities can be derived from agreement of class labels, or by thresholding the original Euclidean distance in \( X \). The second case is multi-level affinities, where \( \mathcal{V} \) consists of non-negative integers. This more fine-grained model of similarity is frequently considered in information retrieval tasks, including in web search engines.

Throughout this paper we assume the setup where a query \( x_q \in X \) is retrieved against some database \( S \subseteq X \). Retrieval is performed by ranking the instances in \( S \) by increasing distance to \( x_q \), using \( d_{\Phi} \) as the distance metric. This is termed “retrieval by Hamming ranking” in the hashing literature. The ranking can be represented by an index vector \( R \), whose elements form a permutation of \( \{1, \ldots, |S|\} \). Below, let \( R_i \) be the \( i \)-th element in \( R \), and \( A_q(i) = \mathcal{A}(x_q, x_i) \). Unless otherwise noted, we implicitly assume dependency on \( x_q, S \), and \( \Phi \) in our notation.

Ranking-based evaluation. Ranking-based metrics usually measure some form of agreement between the ranking and ground truth affinities, capturing the intuition that retrievals with high affinity to the query should be ranked high. First, in the case of binary affinity, we define \( N^+ = |\{x_i \in S | A_q(i) = 1\}| \). Average Precision (AP) averages the precision at cutoff \( k \) over all cutoffs:

\[
\text{AP}(R) = \frac{1}{N^+} \sum_{k=1}^{|S|} A_q(R_k) \left[ \frac{1}{k} \sum_{j=1}^{k} A_q(R_j) \right].
\]

Next, for integer-valued affinities, Discounted Cumulative

\[1\] Although the usual implementation is by counting bit differences, this equivalent formulation has the advantage of being differentiable.

\[2\] The latter is sometimes referred to as “unsupervised hashing” in the literature due to the absence of class labels.
Gain (DCG) is defined as

$$\text{DCG}(R) = \sum_{k=1}^{|S|} G(A_q(R_k)) D(k),$$  \hspace{2cm} (3)

where $$G(a) = 2^a - 1$$, $$D(k) = \frac{1}{\log_2(k+1)}$$  \hspace{2cm} (4)

$$G$$ and $$D$$ are called gain and (logarithmic) discount, respectively. Normalized DCG (NDCG) divides DCG by its maximum possible value, ensuring a range of [0, 1]:

$$\text{NDCG}(R) = \frac{\text{DCG}(R)}{\max_{R'} \text{DCG}(R')}.$$  \hspace{2cm} (5)

### 3.2. Tie-Awareness in Hashing

When evaluating information retrieval systems, special attention is required when there exist ties in the distances [3, 28]. In this case, the ranking $$R$$ is not unique as the items can be ordered arbitrarily, and the tie-breaking strategy may have a sizable impact on the result. We have given an example in Fig. 1. Surprisingly, we found that current ranking-based hashing evaluation protocols usually do not take tie-breaking into account, which could result in ambiguous comparisons or even unfair exploitation. Perhaps more importantly, ties render the formulation of direct optimization unclear: what tie-breaking strategy should we assume when using AP or NDCG as optimization objectives? Thus, we believe that it is important to seek tie-aware evaluation metrics for hashing.

Rather than picking a fixed tie-breaking strategy or relying on randomization, the tie-aware solution that we propose is to average the value of the ranking metric over all possible permutations of tied items. This solution is appealing in several ways: it is deterministic, it is unambiguous and cannot be exploited, and it reduces to the ordinary version when there are no ties. However, there is one caveat: generating all permutations for $$n$$ tied items requires $$O(n!)$$ time, which is super-exponential and prohibitive. Fortunately, [28] observes that the average can be computed implicitly for commonly used ranking metrics, and gives their tie-aware versions in closed form. Based on this result, we further describe how to efficiently compute tie-aware ranking metrics by exploiting the structure of the Hamming distance.

We focus on AP and NDCG, and denote the tie-aware versions of AP and (N)DCG as AP$_T$ and (N)DCG$_T$, respectively. First, we define some notation. With integer-valued Hamming distances, we redefine the ranking $$R$$ to be a collection of $$(b + 1)$$ “ties”, i.e. $$R = \{R^{(0)}, \ldots, R^{(b)}\}$$, where $$R^{(d)} = \{i | d(x_q, x_i) = d\}$$ is the set of retrievals having Hamming distance $$d$$ to the query. We define a set of discrete histograms conditioned on affinity values, $$(n_{0,v}, \ldots, n_{b,v})$$, where $$n_{d,v} = |R^{(d)} \cap \{i | A_q(i) = v\}|$$, $$\forall v \in \mathcal{V}$$, and their cumulative sums $$(N_{0,v}, \ldots, N_{b,v})$$ where $$N_{d,v} = \sum_{j \leq d} n_{j,v}$$. We also define the total histograms as $$n_d = \sum_{v \in \mathcal{V}} n_{d,v}$$ with cumulative sum $$N_d = \sum_{j \leq d} n_j$$.

Next, Proposition 1 gives the closed forms of AP$_T$ and DCG$_T$. We give proof in the appendix.

#### Proposition 1

Both AP$_T$ and DCG$_T$ decompose additively over the ties. For $$\mathcal{V} = \{0, 1\}$$, let $$n^+_d \triangleq n_{d,1}$$, $$N^+_d \triangleq N_{d,1}$$, and $$N^+ = \sum_d n^+_d$$, the contribution of each tie $$R^{(d)}$$ to AP$_T$ is computed as

$$\text{AP}_T(R^{(d)}) = \frac{n^+_d}{n_d N^+} \sum_{t=N_{d-1}+1}^{N_d} \frac{N^+_{d-1} + (t - N_{d-1} - 1) n^+_{d-1} + 1}{t}.$$  \hspace{2cm} (6)

For DCG$_T$, the contribution of $$R^{(d)}$$ is

$$\text{DCG}_T(R^{(d)}) = \sum_{i \in R^{(d)}} \frac{G(A_q(i))}{n_d} \sum_{t=N_{d-1}+1}^{N_d} D(t) = \sum_{v \in \mathcal{V}} \frac{G(v)}{n_d} \sum_{t=N_{d-1}+1}^{N_d} D(t).$$  \hspace{2cm} (7)

**Proof.** See appendix.

---

4025
that |V| is known. We note that counting sort on Hamming distances is also used by Lin et al. [22] to speed up loss-augmented inference for their NDCG surrogate loss.

3.3. The Learning to Rank View

Since we focus on optimizing ranking metrics, our work has connections to learning to rank [24]. Many supervised hashing formulations use loss functions defined on pairs or triplets of training examples, which correspond to pointwise and pairwise approaches in learning to rank terminology. We collectively refer to these as local ranking losses. Since we optimize evaluation metrics defined on a ranked list, our metric is of size tie-aware ranking metrics. For discrete hashing, such optimization is prohibitive to enumerate. Triplet-based methods are also interested in a relaxation approach using gradient-based deep neural networks. Therefore, we apply continuous relaxation to the discrete optimization problems.

4. Optimizing Tie-Aware Ranking Metrics

In this section, we describe our approach to optimizing tie-aware ranking metrics. For discrete hashing, such optimization is NP-hard, since it involves combinatorial search over all configurations of binary bits. Instead, we are interested in a relaxation approach using gradient-based deep neural networks. Therefore, we apply continuous relaxation to the discrete optimization problems.

4.1. Continuous Relaxations

Our continuous relaxation needs to address two types of discrete variables. First, as is universal in hashing formulations, the bits in the hash code are binary. Second, the tie-aware metrics involve integer-valued histogram bin counts \{n_{d,v}\}.

We first tackle the binary bits. Commonly, bits in the hash code are generated by a thresholding operation using the \(\text{sgn}\) function,

\[
\Phi(x) = (\phi_1(x), \ldots, \phi_b(x)),
\]

where in our case \(f_i\) are neural network activations, parameterized by \(w\). We smoothly approximate the \(\text{sgn}\) function using the \(\tanh\) function, which is a standard technique in hashing [4, 8, 20, 25, 41, 42]:

\[
\phi_i(x) \approx \hat{\phi}_i(x) = \text{tanh}(\alpha f_i(x; w)) \in (-1, 1).
\]

The constant \(\alpha\) is a scaling parameter.

As a result of this relaxation, both the hash mapping and the distance function (1) are now real-valued, and will be denoted \(\Phi\) and \(d_{\Phi}\), respectively. The remaining discreteness is from the histogram bin counts \{\(n_{d,v}\)\}. We also relax them into real-valued “soft histograms” \(\{c_{d,v}\}\) (described below), whose cumulative sums are denoted \(\{C_{d,v}\}\). However, we face another difficulty: the definitions of \(\text{AP}_T\) (6) and \(\text{DCG}_T\) (7) both involve a finite sum with lower and upper limits \(N_{d-1} + 1\) and \(N_d\), which themselves are variables to be relaxed. We approximate these finite sums by continuous integrals, removing the second source of discreteness. We outline the results in Proposition 2, and leave proof and error analysis to the appendix.

Importantly, both relaxations have closed-form derivatives. The differentiation for \(\text{AP}_r\) (11) is straightforward, while for \(\text{DCG}_r\) it removes the integral in (12).

4.2. End-to-End Learning

We perform end-to-end learning with gradient ascent. First, as mentioned above, the continuous relaxations \(\text{AP}_r\) and \(\text{DCG}_r\) have closed-form partial derivatives with respect

Proposition 2. The continuous relaxations of \(\text{AP}_T\) and \(\text{DCG}_T\), denoted as \(\text{AP}_r\) and \(\text{DCG}_r\) respectively, are as follows:

\[
\text{AP}_r(R^{(d)}) = \frac{c_d^+ (c_d^+ - 1)}{N^+ (c_d - 1)} + \frac{c_d^+}{N^+ c_d} \left[ (c_d^+)^{c_{d-1}^+} + 1 - \frac{c_d^+ - 1}{c_d - 1} (C_{d-1} + 1) \right] \ln \frac{C_d}{C_{d-1}},
\]

\[
\text{DCG}_r(R^{(d)}) = \ln 2 \sum_{v \in V} \frac{G(v) c_{d,v}}{c_d} \int_{C_{d-1} + 1}^{C_d + 1} \frac{dt}{\ln t}.
\]

Proof. See appendix.

4026
Figure 2: The flow of computation in our model. Input images are mapped to $b$-bit binary codes by a deep neural network ($b = 4$ in this example). During training, in a minibatch, each example is used as query to rank the rest of the batch, producing a histogram of Hamming distances with $(b + 1)$ bins. Tie-aware ranking metrics ($\text{AP}_T$ shown here) are computed on these histograms, and averaged over the batch. To maintain end-to-end differentiability, we derive continuous relaxations for $\text{AP}_T$ and $\text{NDCG}_T$, and employ two differentiable approximations to non-differentiable operations (backward arrows).

to the soft histograms $\{c_{d,v}\}$. Next, we consider differentiating the histogram entries. Note that before relaxation, the discrete histogram $(n_{0,v}, \ldots, n_{b,v})$ for $\forall v \in V$ is constructed as follows:

$$n_{d,v} = \sum_{x_i | A_q(i) = v} 1[d_{\Phi}(x_q, x_i) = d], \; d = 0, \ldots, b.$$  

(13)

To relax $n_{d,v}$ into $c_{d,v}$, we employ a technique from [4,37], where the binary indicator $1[\cdot]$ is replaced by a differentiable function $\delta(d_{\Phi}(x_q, x_i), d)$ with easy-to-compute gradients. Specifically, $\delta(d_{\Phi}(x_q, x_i), d)$ linearly interpolates $d_{\Phi}(x_q, x_i)$ into the $d$-th bin with slope $\Delta > 0$:

$$\forall z \in \mathbb{R}, \delta(z, d) = \begin{cases} 1 - \frac{|z - d|}{\Delta}, & |z - d| \leq \Delta, \\ 0, & \text{otherwise}. \end{cases}$$

(14)

Note that $\delta$ approaches the indicator function as $\Delta \to 0$. We now have the soft histogram $c_{d,v}$ as

$$c_{d,v} = \sum_{x_i | A_q(i) = v} \delta(d_{\Phi}(x_q, x_i), d),$$

(15)

and we differentiate $c_{d,v}$ using chain rule, e.g.

$$\frac{\partial c_{d,v}}{\partial \hat{\Phi}(x_q)} = \sum_{x_i | A_q(i) = v} \frac{\partial \delta(d_{\Phi}(x_q, x_i), d)}{\partial d_{\Phi}(x_q, x_i)} \frac{\partial \hat{\Phi}(x)}{2}.$$  

(16)

The next and final step is to back-propagate gradients to the parameters of the relaxed hash mapping $\hat{\Phi}$, which amounts to differentiating the $\tanh$ function.

As shown in Fig. 2, we train our models using minibatch-based stochastic gradient ascent. Within a minibatch, each example is retrieved against the rest of the minibatch. That is, each example in a minibatch of size $M$ is used as the query $x_q$ once, and participates in the database for some other example $M - 1$ times. Then, the objective is averaged over the $M$ queries.

5. Experiments

5.1. Experimental Setup

We conduct experiments on image retrieval datasets that are commonly used in the hashing literature: CIFAR-10 [16], NUS-WIDE [13], 22K LabelMe [31], and ImageNet100 [8]. Each dataset is split into a test set and a database, and examples from the database are used in training. At test time, queries from the test set are used to perform Hamming ranking on the database, and the performance metric is averaged over the test set.

- **CIFAR-10** is a canonical benchmark for image classification and retrieval, with 60K single-labeled images from 10 classes. Following [42], we consider two experimental settings. In the first setting, the test set is constructed with 100 random images from each class (total: 1K), the rest is used as database, and 500 images per class are used for training (total: 5K). The second setting uses the standard 10K/50K split and the entire database is used in training.

- **NUS-WIDE** is a multi-label dataset with 270K Flickr images. For the database, we use a subset of 196K images associated with the most frequent 21 labels as in [20,42]. 100 images per label are sampled to construct a test set of size 2.1K, and the training set contains 500 images per label (total: 10.5K).

- **LabelMe** is an unlabeled dataset of 22K images. As in [7], we randomly split LabelMe into a test set of size 2K and database of 20K. We sample 5K examples from the database for training.

- **ImageNet100** is a subset of ImageNet, containing all the images from 100 classes. We use the same setup as in [8]: 130 images per class, totaling 130K images, are sampled for training, and all images in the selected classes from the ILSVRC 2012 validation set are used as queries.
Retrieval-based evaluation of supervised hashing was recently put into question by [32], which points out that for multi-class datasets, binary encoding of classifier outputs is already a competitive solution. While this is an important point, deriving pairwise affinities from multi-class label agreement is a special case in our formulation. As mentioned in Sec. 3.1, our formulation uses a general pairwise affinity oracle $A$, which may or may not be derived from labels, and can be either binary or multi-level. In fact, the datasets we consider range from multi-class/single-label (CIFAR-10, NUS-WIDE, and ImageNet100) to multi-label (NUS-WIDE) and unlabeled (LabelMe), and only the first case can be addressed by multi-class classification. For multi-level affinities, we also propose a new evaluation protocol using NDCG.

We term our method TALR (Tie-Aware Learning to Rank), and compare it against a range of classical and state-of-the-art hashing methods. Due to the vast hashing literature, an exhaustive comparison is unfortunately not feasible. Focusing on the learning to rank aspect, we select representative methods from all three categories:

- **Pointwise (pair-based).** Methods that define loss functions on instance pairs: Binary Reconstructive Embeddings (BRE) [18], Fast Supervised Hashing (FastHash) [23], Hashing using Auxiliary Coordinates (MACHash) [30], Deep Pair-Supervised Hashing (DPSH) [20], and Hashing by Continuation (HashNet) [8].

- **Pairwise (triplet-based).** We include a recent method, Deep Triplet-Supervised Hashing (DTSH) [42].

- **Listwise (list-based).** We compare to two listwise ranking methods: Structured Hashing (StructHash) [22] which optimizes an NDCG surrogate, and Hashing with Mutual Information (MIHash) [4] which optimizes mutual information as a ranking surrogate for binary affinities.

These selected methods include recent ones that achieve state-of-the-art results on CIFAR-10 (MIHash, DTSH), NUS-WIDE (DTSH, HashNet) and ImageNet100 (HashNet).

Since tie-aware evaluation of Hamming ranking performance has not been reported in the hashing literature, we re-train and evaluate all methods using publicly available implementations.

### 5.2.2 ImageNet100

For ImageNet100 experiments, we closely follow the setup in HashNet [8] and fine-tune the AlexNet architecture [17] pretrained on ImageNet. Due to space limitations, we report comparisons against recent state-of-the-art methods on ImageNet100. The first competitor is HashNet, which is empirically superior to a wide range of classical and recent methods, and was previously the state-of-the-art method on ImageNet100. We also compare to MIHash, as it is the second-best method on CIFAR-10 and NUS-WIDE in the previous experiment. As in [8], the minibatch size is set to 256 for all methods, and the learning rate for the pre-trained convolution and fully connected layers are scaled down, since the model is fine-tuned on the same dataset that it was originally trained on. AP at cutoff 1000 (AP@1000) is used as the evaluation metric.

ImageNet100 results are summarized in Table 2. TALR-AP outperforms both competing methods, and the improvement is especially significant with short hash codes (16 and 32 bits). This indicates that our direct optimization approach produces better compact binary representations that preserve desired rankings. The state-of-the-art performance with com-
T ALR-AP outperforms state-of-the-art solutions using multi-level affinities within the top 5% of the desired structure of the learned Hamming space. To our knowledge, this setup has not been considered before. Inspired by an existing binary affinity setup \([7]\) that defines neighbors as having Euclidean distances between examples, we consider the set of affinity values \(V\) as a finite set of non-negative integers. Multi-level affinities are common in information retrieval tasks, and offer more fine-grained specificity of the desired structure of the learned Hamming space. To our knowledge, this setup has not been considered in the hashing literature.

In the multi-label NUS-WIDE dataset, we define the affinity value between two examples as the number of labels they share, and keep other settings the same as in the AP experiment. For the unlabeled LabelMe dataset, we derive affinities by thresholding the Euclidean distances between examples. Inspired by an existing binary affinity setup \([7]\) that defines neighbors as having Euclidean distances within the top 5% on the training set, we use four thresholds \(\{5\%, 1\%, 0.2\%, 0.1\%\}\) and assign affinity values \(\{1, 2, 5, 10\}\). This emphasizes assigning high ranks to the closest neighbors in the original feature space. We learn shallow models on precomputed GIST features on LabelMe. For gradient-based methods, this means using linear hash functions, \(i.e. f_i(x; w) = w_i^\top x\), in \((9)\). For methods that are not designed to use multi-level affinities (FastHash, MACHash, DPSH, MIHash), we convert the affinities into binary values; this reduces to the standard binary affinity setup on both datasets.

We lastly discuss the effect of tie-breaking in evaluating hashing algorithms. As mentioned in Sec. 3.2, tie-breaking is an uncontrolled parameter in current evaluation protocols, which can affect results, and even be exploited. To demonstrate this, we consider for example the AP experiment in CIFAR-10’s first setting, presented in Sec. 5.2. For each

<table>
<thead>
<tr>
<th>Method</th>
<th>CIFAR-10</th>
<th>NUS-WIDE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>16 Bits</td>
<td>32 Bits</td>
</tr>
<tr>
<td>BRE [18]</td>
<td>0.361</td>
<td>0.448</td>
</tr>
<tr>
<td>MACHash [30]</td>
<td>0.628</td>
<td>0.707</td>
</tr>
<tr>
<td>FastHash [23]</td>
<td>0.678</td>
<td>0.729</td>
</tr>
<tr>
<td>StructHash [22]</td>
<td>0.664</td>
<td>0.693</td>
</tr>
<tr>
<td>DPSH [20]*</td>
<td>0.720</td>
<td>0.757</td>
</tr>
<tr>
<td>DTSH [42]</td>
<td>0.725</td>
<td>0.773</td>
</tr>
<tr>
<td>MIHash [4]</td>
<td>0.687</td>
<td>0.775</td>
</tr>
<tr>
<td>TALR-AP</td>
<td>0.732</td>
<td>0.789</td>
</tr>
</tbody>
</table>

* Trained using parameters recommended by authors of DTSH.

Table 1: AP comparison on CIFAR-10 and NUS-WIDE with VGG-F architecture. On CIFAR-10, we compare all methods in the first setting (S1), and deep learning methods in the second (S2). We report the tie-aware AP, and additionally AP@5K for NUS-WIDE. TALR-AP optimizes tie-aware AP using stochastic gradient ascent, and achieves state-of-the-art performance.

<table>
<thead>
<tr>
<th>Method</th>
<th>16 Bits</th>
<th>32 Bits</th>
<th>48 Bits</th>
<th>64 Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>HashNet [8]</td>
<td>0.5059</td>
<td>0.6306</td>
<td>0.6633</td>
<td>0.6835</td>
</tr>
<tr>
<td>MIHash [4]</td>
<td>0.5688</td>
<td>0.6608</td>
<td>0.6852</td>
<td>0.6947</td>
</tr>
<tr>
<td>TALR-AP</td>
<td>0.5892</td>
<td>0.6689</td>
<td>0.6985</td>
<td>0.7053</td>
</tr>
</tbody>
</table>

Table 2: AP@1000 results on ImageNet100 with AlexNet. TALR-AP outperforms state-of-the-art solutions using mutual information [4] and continuation methods [8].
Table 3: NDCG comparison on NUS-WIDE (VGG-F architecture) and LabelMe (shallow models on GIST features). TALR-NDCG optimizes tie-aware NDCG using stochastic gradient ascent, and consistently outperforms competing methods.

![Image](image.png)

Figure 3: Effects of tie-breaking: we plot the ranges of test-time mAP values spanned by all possible tie-breaking strategies, for all methods considered in the CIFAR-10 experiment (first setting). Horizontal axis: mAP. Black dots: values of tie-aware AP. Without controlling for tie-breaking, relative performance comparison between different methods can be ambiguous. The ambiguity is eliminated by tie-awareness.

We note that although random tie-breaking is an approximation to tie-awareness at test time, it does not answer the question of how to optimize the ranking metrics during training. Our original motivation is to optimize ranking metrics for hashing, and the existence of closed-form tie-aware ranking metrics is what makes direct optimization feasible.

6. Conclusion

We have proposed a new approach to hashing for nearest neighbor retrieval, with an emphasis on directly optimizing evaluation metrics used at test-time. A study into the commonly used retrieval by Hamming ranking setup led us to consider the issue of ties, and we advocate for using tie-aware versions of ranking metrics. We then make the novel contribution of optimizing tie-aware ranking metrics for hashing, focusing on the important special cases of AP and NDCG. To tackle the resulting discrete and NP-hard optimization problems, we derive their continuous relaxations. Then, we perform end-to-end stochastic gradient ascent with deep neural networks. This results in the new state-of-the-art for common image retrieval benchmarks.

Acknowledgements

The authors would like to thank Qinxun Bai, Peter Gacs, and Dora Erdos for helpful discussions. This work is supported in part by a BU IGNITION award, NSF grant 1029430, and gifts from Nvidia.
References


