A. Proof of Proposition 1

Proof. Our proof essentially restates the results in [3] using our notation. In [3], a tie-vector \( T = (t_0, \ldots, t_{d+1}) \) is defined, where \( t_0 = 0 \) and the next elements indicate the ending indices of the equivalence classes in the ranking, e.g. \( t_1 \) is the ending index of \( R^{(0)} \), and so on. Using our notation, we can see that \( R^{(d)} = (R_{1+t_d}, \ldots, R_{t_{d+1}}) \), and \( t_d = N_{d-1} = \sum_{j=0}^{d-1} n_j \).

We first consider \( \text{AP}_T \). In Section 2.4 of [3], tie-aware \( \text{AP} \) is defined as

\[
\text{AP}_T(R) = \frac{1}{N^+} \sum_{j=1}^{|S|} \frac{n_i^+}{n_i} \left( N_{i-1}^+ + (j - t_i - 1) \frac{n_i^+-1}{n_i-1} + 1 \right) \frac{1}{j},
\]

where \( i \) is the index of the tie that item \( j \) is in. To derive \( \text{AP}_T \) in our formulation, we take \( k \) to be the maximum possible cutoff \( |S| \), and substitute by definition \( N^+ = \sum_{j=1}^{|S|} A_q(j), t_i = N_{i-1} \):

\[
\text{AP}_T(R) = \frac{1}{N^+} \sum_{j=1}^{|S|} \frac{n_i^+}{n_i} \left( N_{i-1}^+ + (j - N_{i-1} - 1) \frac{n_i^+-1}{n_i-1} + 1 \right) \frac{1}{j}.
\]

It is clear that this sum decomposes additively over \( j \). Therefore, we can explicitly compute the contribution from items in a single tie \( R^{(d)} \),

\[
\text{AP}_T(R^{(d)}) = \frac{1}{N^+} \sum_{j=N_{d-1}+1}^{N_d} \frac{n_i^+}{n_i} \left( N_{d-1}^+ + (j - N_{d-1} - 1) \frac{n_i^+-1}{n_i-1} + 1 \right) \frac{1}{j},
\]

and this gives (6) in the paper.

Next, tie-aware DCG is given in Section 2.6 of [3] as

\[
\text{DCG}_T(R) = \sum_{k=1}^{t_{d+1}} \left( \frac{1}{n_d} \sum_{j=t_{d+1}+1}^{t_{d+1}} G(A_q(j)) \right) \sum_{j=t_{d+1}+1}^{\min(t_{d+1}, k)} D(j)
\]

Again, we consider a single tie \( R^{(d)} \), take \( k = |S| \), and make the substitution \( t_d = N_{d-1} \):

\[
\text{DCG}_T(R^{(d)}) = \frac{1}{n_d} \sum_{j \in R^{(d)}} G(A_q(j)) \sum_{j=N_{d-1}+1}^{N_d} D(j) \tag{5}
\]

\[
= \frac{1}{n_d} \sum_{v \in V} \sum_{j \in R^{(d)}} 1[v = A_q(j)] G(v) \sum_{j=N_{d-1}+1}^{N_d} D(j) \tag{6}
\]

\[
= \frac{1}{n_d} \sum_{v \in V} G(v) n_d, v \sum_{j=N_{d-1}+1}^{N_d} D(j). \tag{7}
\]

This completes the derivation for (7) in the paper.
B. Proof of Proposition 2

Proof. First, we denote the summand in the definition of AP_T (3) as \( \beta_d(t) \), and rewrite it as

\[
\beta_d(t) = \frac{N_d^+ + (t - N_{d-1} - 1)\frac{n_{d-1}^+ - 1}{n_d} + 1}{t} = \frac{n_d^+}{n_{d-1}} - 1 + \frac{N_{d-1}^+ + 1 - \frac{n_{d-1}^+}{n_d}}{t}. \tag{8}
\]

It is of the form \( A + B/t \) where \( A, B \) are constant in \( t \). We proceed with the summation over \( t \) in (3):

\[
AP_T(R^{(d)}) = \frac{n_d^+}{n_{d+}N^+} \sum_{t=N_{d-1}+1}^{N_d} \beta_d(t)
= \frac{n_d^+}{n_{d+}N^+} \left[ \frac{n_d^+}{n_{d-1}} - 1 + \left( N_{d-1}^+ + 1 - \frac{n_{d-1}^+}{n_d} \right) \sum_{t=N_{d-1}+1}^{N_d} \frac{1}{t} \right]. \tag{10}
\]

The main obstacle in the continuous relaxation is the finite sum in (10), which has in its limits \( N_{d-1} \) and \( N_d \), variables to be relaxed. However, it is a partial sum of the harmonic series, which can be well approximated by differences of the natural logarithm:

\[
\sum_{t=N_{d-1}+1}^{N_d} \frac{1}{t} \approx \int_{N_{d-1}}^{N_d} \frac{dt}{t} = \ln(N_d) - \ln(N_{d-1}). \tag{11}
\]

In fact, (11) corresponds to the midpoint rule in approximating definite integrals by finite sums, but is applied in the reverse direction. The relaxation of AP_T is then derived as follows:

\[
AP_T(R^{(d)}) \approx \frac{n_d^+}{n_{d+}N^+} \left[ \frac{n_d^+}{n_{d-1}} - 1 + \left( N_{d-1}^+ + 1 - \frac{n_{d-1}^+}{n_d} \right) \ln \frac{N_d}{N_{d-1}} \right]
\Rightarrow AP_T(R^{(d)}) = \frac{c_d^+(c_d^+ - 1)}{(c_d - 1)N^+} + \frac{c_d^+}{c_{d+}N^+} \left[ C_{d-1}^+ + 1 - \frac{c_d^+}{c_d - 1} \ln \frac{C_d}{C_{d-1}} \right]. \tag{12}
\]

Note that \( N^+ = \sum_d n_d^+ \) is a constant for a fixed query and fixed database, thus it is not affected by the relaxation.

Next, we consider DCG_T, where the sum of logarithmic\(^1\) discount values similarly involves variables to be relaxed in its limits. Thus, we employ the same approximation strategy using continuous integrals.

\[
\sum_{t=N_{d-1}+1}^{N_d} D(t) = \sum_{t=N_{d-1}+1}^{N_d} \frac{1}{\log_2(t + 1)} \approx \int_{N_{d-1}}^{N_d} \frac{dt}{\log_2(t + 1)} = \ln 2 \int_{N_{d-1}+1}^{N_d+1} \frac{dt}{\ln t}. \tag{14}
\]

Combining with the definition of DCG_T, we get its continuous relaxation:

\[
DCG_T(R^{(d)}) = \ln 2 \sum_{v \in V} \frac{G(v)c_{d,v}}{c_d} \int_{C_{d-1}+1}^{C_d+1} \frac{dt}{\ln t}
= \ln 2 \sum_{v \in V} \frac{G(v)c_{d,v}}{c_d} \left[ Li(C_d + 1) - Li(C_{d-1} + 1) \right] \tag{15}
\]

where \( Li \) is the logarithmic integral function: \( Li(x) = \int_0^x \frac{dx}{\ln x} \).

C. Approximation Error Analysis

We now analyze the approximation error when doing the continuous relaxations. We take AP_T as example, and note that the analysis for DCG_T is similar.

\(^1\)Other types of discounts are also used in the literature, including linear discount: \( D(t) \propto \frac{1}{t} \). It is easy to see that our technique also applies.
The continuous relaxation for $AP_T(R^{(d)})$ is given in (11), which replaces a finite sum with a definite integral, where the finite sum has $N_d - N_{d-1} = n_d$ summands. First, we consider the case where there are no ties, or $n_d \in \{0,1\}$, i.e. the $d$-th histogram bin is either empty or contains a single item. In this case, we can directly evaluate the lefthand side sum in (11) to be either 0 or $\frac{1}{N_d}$, without using the integral approximation. Therefore, when there are no ties, there is no approximation error.

Next we consider $n_d \geq 2$. Let the $N$-th harmonic number be $H(N) = \sum_{t=1}^{N} \frac{1}{t}$, then the lefthand side of (11) is exactly $H(N_d) - H(N_{d-1})$. It is well known that the harmonic number can be closely approximated as

$$H(N) = \gamma + \ln(N) + \frac{1}{2N} + O\left(\frac{1}{12N^2}\right),$$

where $\gamma \approx 0.5772$ is Euler’s constant. A direct application of this approximation gives the following:

$$H(N_d) = \gamma + \ln(N_d) + \frac{1}{2N_d} + O\left(\frac{1}{12N_d^2}\right)$$

$$H(N_{d-1}) = \gamma + \ln(N_{d-1}) + \frac{1}{2N_{d-1}} + O\left(\frac{1}{12N_{d-1}^2}\right)$$

$$\Rightarrow H(N_d) - H(N_{d-1}) = \ln(N_d) - \ln(N_{d-1}) + O\left(\frac{1}{2N_{d-1}} - \frac{1}{2N_d}\right).$$

Comparing (20) with (11), we see that the approximation error is

$$O\left(\frac{1}{2N_{d-1}} - \frac{1}{2N_d}\right) = O\left(\frac{n_d}{2N_{d-1}N_d}\right) = O\left(\frac{n_d}{2N_{d-1}(N_{d-1} + n_d)}\right) = O\left(\frac{n_d}{2N_{d-1}^2}\right).$$

The error is proportional to $n_d$, the number of items in the $d$-th bin in the Hamming distance histogram. However, even if $n_d$ is large, the error is in general still small, since it has $N^2_{d-1}$ in the denominator. Note that (11) can be further tightened by including the $\frac{1}{N}$ term, or even higher order terms in the approximation of Harmonic numbers, but the approximation using the first two terms (Euler’s constant and natural log) is already quite tight, and is in fact used widely.

**D. Tightening the tanh Relaxation**

As is common among relaxation-based hashing methods, we relax the binary constraints by replacing the discontinuous $\text{sgn}$ function with $\tanh$. With this simple relaxation, the performance gains are mainly due to optimizing the proposed objectives. Nevertheless, it is conceivable that the continuation method, e.g. as employed by HashNet [2], can tighten the relaxation and lead to better results. To provide a concrete example, below we report improved results for TALR-AP on CIFAR-10 (setting 1), when we increase the scaling $\alpha$ in tanh over time, instead of using a fixed value.

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>12 bits</th>
<th>24 bits</th>
<th>32 bits</th>
<th>48 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>0.752</td>
<td>0.789</td>
<td>0.800</td>
<td>0.826</td>
</tr>
<tr>
<td>$\to\infty$</td>
<td>0.751</td>
<td>0.804</td>
<td>0.813</td>
<td>0.830</td>
</tr>
</tbody>
</table>

**E. Efficient Minibatch Backpropagation**

As mentioned in Sec. 4.2 in the paper, our models are trained using minibatch SGD. To maximally utilize supervision, we use the following strategy: each example in the minibatch is used to query the rest of the batch (which acts as the database), and the resulting objective values are averaged. Here, we detail the derivation of the backpropagation rules according to this formulation. We take inspirations from [1] and [4], both of which similarly average listwise objectives designed for binary affinities. Our technique can be seen as a direct generalization of [1] in that we support multi-level affinities.

Consider a minibatch of size $M$, $\{x_1, \ldots, x_M\}$. We use a unified shorthand $O_i$ to denote the (relaxed) objective value when $x_i$ is the query, which can either be $AP_T$ or $\text{DCG}_i$ in our formulation. The overall minibatch objective is then $O = \frac{1}{M} \sum_{i=1}^{M} O_i$. For the entire minibatch, we group the relaxed hash mapping output into a $b \times M$ matrix,

$$\hat{\Phi} = \left[ \hat{\Phi}(x_1) \hat{\Phi}(x_2) \cdots \hat{\Phi}(x_M) \right] \in \mathbb{R}^{b \times M}.$$  

(22)
We consider the multi-level affinity setup where affinity values are from a finite set \( V \), which includes binary affinities as a special case, \( i.e. V = \{0, 1\} \). The Jacobian of the minibatch objective with respect to \( \Phi \) can be written as
\[
\frac{\partial O}{\partial \Phi} = \frac{1}{M} \sum_{i=1}^{M} \frac{\partial O_i}{\partial \Phi} = \frac{1}{M} \sum_{i=1}^{M} \sum_{d=0}^{b} \sum_{v \in V} \frac{\partial O_i}{\partial c_{d,v}^{(i)}} \frac{\partial c_{d,v}^{(i)}}{\partial \Phi},
\]
where as defined in Sec. 4.1, \( c_{d,v}^{(i)} \) is the continuous relaxation of \( n_{d,v}^{(i)} \), the \( d \)-th bin in the distance histogram conditioned on affinity level \( v \). The superscript \( (i) \) indicates that the current query is \( x_i \).

Evaluating this Jacobian involves two steps. First, we need to compute the partial derivative \( \partial O_i / \partial c_{d,v}^{(i)} \), \( \forall d, \forall v \). Note that this is exactly the differentiation of AP\(_r\), and DCG\(_r\), and as we pointed out in Sec. 4.1, both can be evaluated in closed form. We use variables \( \alpha \) to denote these partial derivatives, omitting the details of derivation:
\[
\alpha_{d,v}^{(i)} = \frac{\partial O_i}{\partial c_{d,v}^{(i)}},
\]
(24)

Next, we need to evaluate the Jacobian \( \partial c_{d,v}^{(i)} / \partial \Phi \), which is essentially differentiating the soft histogram binning process. Let us consider each column of this Jacobian. First, for \( \forall j \neq i \), using chain rule,
\[
\frac{\partial c_{d,v}^{(i)}}{\partial \Phi(x_j)} = \frac{\partial c_{d,v}^{(i)}}{\partial d_{\Phi}(x_i, x_j)} \frac{\partial d_{\Phi}(x_i, x_j)}{\partial \Phi(x_j)} = 1[A_i(j) = v] \delta[d_{\Phi}(x_i, x_j), d] \frac{\partial \hat{d}_{\Phi}(x_i, x_j)}{\partial \Phi(x_j)}
\]
(25)
\[
= 1[A_i(j) = v] \delta_d'(\hat{d}_{\Phi}(x_i, x_j)) \frac{-\hat{\Phi}(x_i)}{2}
\]
(26)
\[
\Delta = \beta_{d,v}(i, j) \frac{-\hat{\Phi}(x_i)}{2},
\]
(27)
where \( \delta_d' \) is the derivative of the single-argument function \( \delta(\cdot, d) \), and we define the shorthand
\[
\beta_{d,v}(i, j) = 1[A_i(j) = v] \delta_d'(\hat{d}_{\Phi}(x_i, x_j)).
\]
(28)

Note that \( \beta \) is symmetric, \( i.e. \beta_{d,v}(i, j) = \beta_{d,v}(j, i) \), which follows from the fact that both the affinity \( A \) and the distance function \( d_{\Phi} \) are symmetric.

For \( j = i \), we have a similar result:
\[
\frac{\partial c_{d,v}^{(i)}}{\partial \Phi(x_i)} = \sum_{k \neq i} \frac{\partial c_{d,v}^{(i)}}{\partial d_{\Phi}(x_i, x_k)} \frac{\partial \hat{d}_{\Phi}(x_i, x_k)}{\partial \Phi(x_i)} = \sum_{k \neq i} \beta_{d,v}(i, k) \frac{-\hat{\Phi}(x_k)}{2}.
\]
(29)

To unify these two cases, we require that \( \beta_{d,v}(i, i) \equiv 0, \forall i \). We now have a unified expression for the \( j \)-th column in the Jacobian \( \partial c_{d,v}^{(i)} / \partial \Phi \):
\[
\frac{\partial c_{d,v}^{(i)}}{\partial \Phi(x_j)} = -\frac{1}{2} \beta_{d,v}(i, j) \hat{\Phi}(x_i) - \frac{1}{2} \sum_{k=1}^{M} \beta_{d,v}(i, k) \hat{\Phi}(x_k).
\]
(30)

We now obtain a compact matrix form for \( \partial c_{d,v}^{(i)} / \partial \Phi \). Let \( \beta_{d,v}^{(i)} = (\beta_{d,v}(i, 1), \ldots, \beta_{d,v}(i, M)) \in \mathbb{R}^M \). Also, let \( e_i \) be the \( i \)-th standard basis vector in \( \mathbb{R}^M \), \( i.e. \) the \( i \)-th element is 1 and all others are 0. We have that
\[
\frac{\partial c_{d,v}^{(i)}}{\partial \Phi} = -\frac{1}{2} \hat{\Phi}(x_i)(\beta_{d,v}^{(i)})^\top - \left[ \sum_{k=1}^{M} \frac{1}{2} \beta_{d,v}(i, k) \hat{\Phi}(x_k) \right] e_i^\top = -\frac{1}{2} \left[ \hat{\Phi}(x_i)(\beta_{d,v}^{(i)})^\top + \hat{\Phi}(\beta_{d,v}^{(i)})^\top e_i \right].
\]
(31)
Finally, we complete (23) using the result above. The main trick is to change the ordering of sums: we bring the sum over $i = 1, \ldots, M$ inside,

$$
\frac{\partial \mathcal{O}}{\partial \hat{\Phi}} = \frac{1}{M} \sum_{i=1}^{M} \sum_{d=0}^{b} \sum_{v \in \mathcal{V}} \frac{\partial c^{(i)}_{d,v}}{\partial \hat{\Phi}}
$$

$$
= \frac{1}{M} \sum_{d=0}^{b} \sum_{v \in \mathcal{V}} \sum_{i=1}^{M} \frac{1}{2} \alpha^{(i)}_{d,v} \left[ \hat{\Phi}(x_i)(\beta^{(i)}_{d,v})^\top + \hat{\Phi}_d \beta^{(i)}_{d,v} e_i^\top \right]
$$

$$
= -\frac{1}{2M} \sum_{l=0}^{b} \sum_{v \in \mathcal{V}} \left[ \sum_{i=1}^{M} \alpha^{(i)}_{d,v} \hat{\Phi}(x_i)(\beta^{(i)}_{d,v})^\top + \hat{\Phi}_d \sum_{i=1}^{M} \alpha^{(i)}_{d,v} \beta^{(i)}_{d,v} e_i^\top \right].
$$

To further simplify this result, we define two $M \times M$ matrices:

$$
A_{d,v} = \text{diag}(\alpha^{(1)}_{d,v}, \ldots, \alpha^{(M)}_{d,v}) \in \mathbb{R}^{M \times M},
$$

$$
B_{d,v} = \begin{bmatrix}
\beta^{(1)}_{d,v} & \cdots & \beta^{(M)}_{d,v}
\end{bmatrix} \in \mathbb{R}^{M \times M}.
$$

Then, we arrive at the following simplification of (34) and (23),

$$
\frac{\partial \mathcal{O}}{\partial \hat{\Phi}} = -\frac{1}{2M} \sum_{d=0}^{b} \sum_{v \in \mathcal{V}} \left[ \hat{\Phi} A_{d,v} (B_{d,v})^\top + \hat{\Phi}_d B_{d,v} A_{d,v} \right] = -\frac{\hat{\Phi}}{2M} \sum_{d=0}^{b} \sum_{v \in \mathcal{V}} \left( A_{d,v} B_{d,v} + B_{d,v} A_{d,v} \right).
$$

Note that we have used the fact that $B_{d,v}$ is a symmetric matrix (36), which is because $\beta$ is symmetric, as mentioned earlier. This operation can be implemented efficiently using only matrix multiplications and additions. Also, since $A_{d,v}$ is a diagonal matrix, multiplying it with $B_{d,v}$ essentially scales the rows or columns of $B_{d,v}$, which is an $O(M^2)$ operation as opposed to $O(M^3)$ as in general matrix multiplication. The entire time complexity is therefore $O(b|\mathcal{V}|M^2)$.

At this point, we have completed the differentiation of the minibatch objective $\mathcal{O}$ with respect to the relaxed hash mapping output, $\hat{\Phi}$. Further backpropagation is straightforward, since $\hat{\Phi}$ is obtained by applying a pointwise tanh function on the raw activations from the previous layer.

References


