Hashing as Tie-Aware Learning to Rank

Kun He, Fatih Cakir, Sarah Adel Bargal, Stan Sclaroff
Computer Science, Boston University

BOSTON UNIVERSITY

2018 CVPR SALT LAKE CITY • JUNE 18-22

Hashing: Learning to Optimize AP / NDCG

- Hashing: binary embeddings for fast Nearest Neighbors
- Evaluation: Average Precision (AP), Normalized Discounted Cumulative Gain (NDCG), etc.

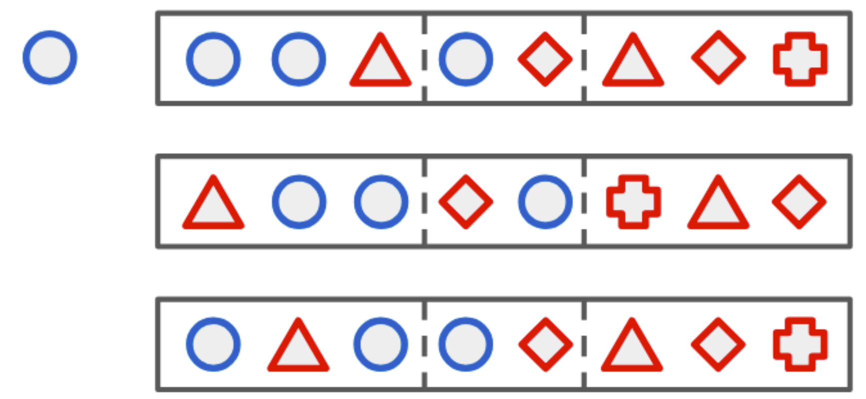
Affinity: 1 3 0 2 0 0 0 0

$$\mathrm{AP} = \mathrm{avg}\left(\frac{1}{1}\quad \frac{2}{2}\quad \frac{3}{4}\right) = 0.92$$

$$\mathrm{NDCG} = \mathrm{sum}\left(\frac{2^1}{\log(1+1)}\quad \frac{2^3}{\log(2+1)}\quad \frac{2^2}{\log(4+1)}\right) / \mathrm{maxDCG} = 0.76$$

- Problem: with Hamming distance, ranking has ties

How should we break the ties?

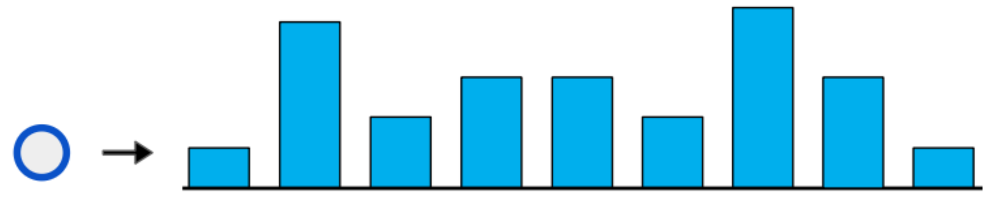


- Tie-aware ranking metrics [1]: average over all permutations of tied items, in **closed-form**

$$\mathrm{AP_T}(R^{(d)}) = \frac{n_d^+}{n_d N^+}\sum_{t=N_{d-1}+1}^{N_d}\frac{N_{d-1}^+ + (t-N_{d-1}-1)\frac{n_d^+-1}{n_d-1}+1}{t}$$

$$\mathrm{DCG_T}(R^{(d)}) = \sum_{i\in R^{(d)}}\frac{G(\mathcal{A}_q(i))}{n_d}\sum_{t=N_{d-1}+1}^{N_d}D(t) = \sum_{v\in\mathcal{V}}\frac{G(v)n_{d,v}}{n_d}\sum_{t=N_{d-1}+1}^{N_d}D(t)$$

- Key operation:
Histogram binning



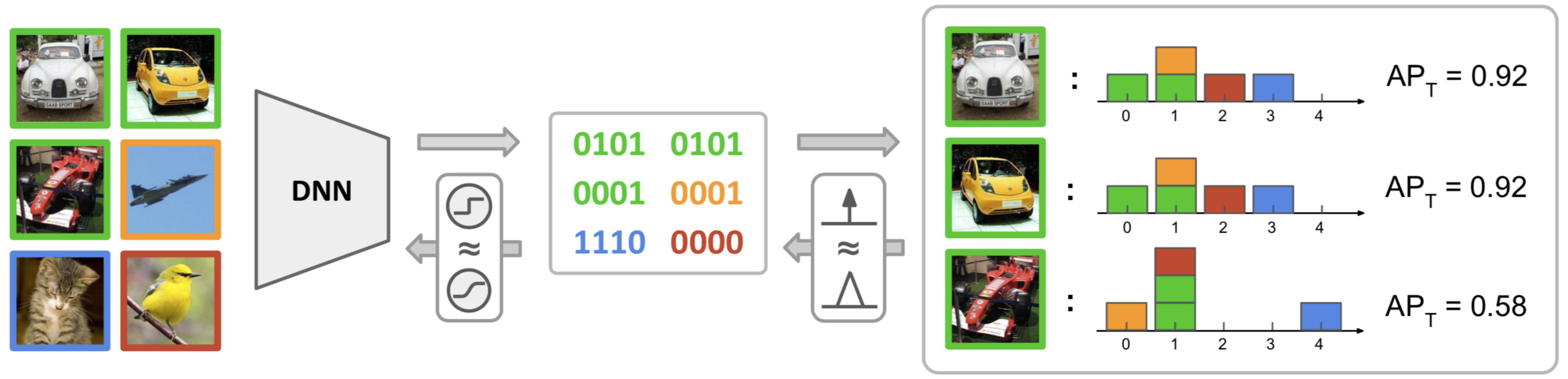
Optimizing Tie-Aware AP / NDCG

- Optimization: combinatorial, NP-hard (search over all bit combinations)
- To apply SGD: continuous relaxation!
- Main trick: reverse "midpoint rule": finite sums → continuous integrals

$$\mathrm{AP_r}(R^{(d)}) = \frac{c_d^+(c_d^+-1)}{N^+(c_d-1)} + \frac{c_d^+}{N^+c_d}\left[C_{d-1}^+ + 1 - \frac{c_d^+-1}{c_d-1}(C_{d-1}+1)\right]\ln\frac{C_d}{C_{d-1}},$$

$$\mathrm{DCG_r}(R^{(d)}) = \ln 2\sum_{v\in\mathcal{V}}\frac{G(v)c_{d,v}}{c_d}\int_{C_{d-1}+1}^{C_d+1}\frac{dt}{\ln t}.$$

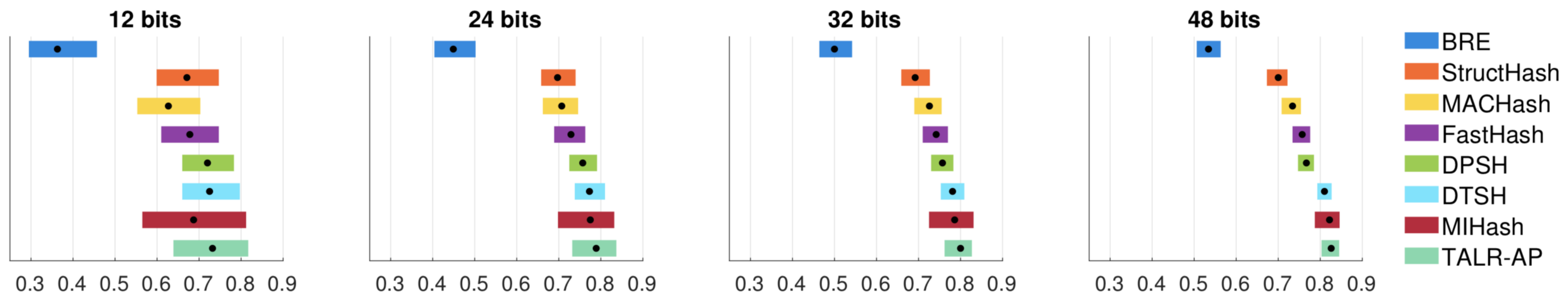
- End-to-end optimization (closed-form gradients)



- Relaxing the binary constraint: $\mathbf{1}[x>0] \approx \tanh(x)$
- Differentiable histogram binning [2]

Tie-Breaking Matters

- No tie-awareness in **training**: optimization objective unclear!
- No tie-awareness in **testing**: ambiguity in results, even exploitation!



Experiments

- Image retrieval by Hamming ranking, VGG-F architecture
- Binary affinity (metric: AP)

Setting	Method	CIFAR-10 12 Bits	CIFAR-10 24 Bits	CIFAR-10 32 Bits	CIFAR-10 48 Bits	NUS-WIDE 12 Bits	NUS-WIDE 24 Bits	NUS-WIDE 32 Bits	NUS-WIDE 48 Bits	Metric
S1 ($\mathrm{AP_T}$)	BRE [18]	0.361	0.448	0.502	0.533	0.561	0.578	0.589	0.607	$\mathrm{AP_T}$
	MACHash [30]	0.628	0.707	0.726	0.734	0.361	0.361	0.361	0.361	
	FastHash [23]	0.678	0.729	0.742	0.757	0.646	0.686	0.698	0.712	
	StructHash [22]	0.664	0.693	0.691	0.700	0.639	0.645	0.666	0.669	
	DPSH [20]*	0.720	0.757	0.757	0.767	0.658	0.674	0.695	0.700	
	DTSH [42]	0.725	0.773	0.781	0.810	0.660	0.700	0.707	0.723	
	MIHash [4]	0.687	0.775	0.786	0.822	0.652	0.693	0.709	0.723	
	TALR-AP	**0.732**	**0.789**	**0.800**	**0.826**	**0.709**	**0.734**	**0.745**	**0.752**	
	Method	16 Bits	24 Bits	32 Bits	48 Bits	12 Bits	24 Bits	32 Bits	48 Bits	
S2 ($\mathrm{AP_T}$)	DPSH [20]*	0.908	0.909	0.917	0.932	0.758	0.793	0.818	0.830	AP@5K
	DTSH [42]	0.916	0.924	0.927	0.934	0.773	0.813	0.820	0.838	
	MIHash [4]	0.929	0.933	0.938	0.942	0.767	0.784	0.809	0.834	
	TALR-AP	**0.939**	**0.941**	**0.943**	**0.945**	**0.795**	**0.835**	**0.848**	**0.862**	

- Multi-level affinity (metric: NDCG)

Method	NUS-WIDE 16 Bits	NUS-WIDE 32 Bits	NUS-WIDE 48 Bits	NUS-WIDE 64 Bits	LabelMe 16 Bits	LabelMe 32 Bits	LabelMe 48 Bits	LabelMe 64 Bits
BRE [18]*	0.805	0.817	0.827	0.834	0.807	0.848	0.871	0.880
MACHash [30]	0.821	0.821	0.821	0.821	0.683	0.683	0.683	0.687
FastHash [23]	0.885	0.896	0.899	0.902	0.844	0.868	0.855	0.864
DPSH [20]	0.895	0.905	0.909	0.909	0.844	0.856	0.871	0.874
DTSH [42]	0.896	0.905	0.911	0.913	0.838	0.852	0.859	0.862
StructHash [22]	0.889	0.893	0.894	0.898	0.857	0.888	0.904	0.915
MIHash [4]	0.886	0.903	0.909	0.912	0.860	0.889	0.907	0.914
TALR-NDCG	**0.903**	**0.910**	**0.916**	**0.927**	**0.866**	**0.895**	**0.908**	**0.917**

References

[1] F. McSherry, M. Najork. Computing Information Retrieval Performance Measures Efficiently in the Presence of Tied Scores, ECIR 2008.

[2] E. Usnitova, V. Lempitsky. Learning Deep Embeddings with Histogram Loss, NIPS 2016.

http://github.com/kunhe/TALR