NEURAL INFORMATION PROCESSING SYSTEMS 2021

One Loss for All: Deep Hashing with a Single Cosine Similarity based Learning Objective

Jiun Tian Hoe[1]*, Kam Woh Ng[2,3]*, Tianyu Zhang[4]
Chee Seng Chan[1], Yi-Zhe Song[2,3] & Tao Xiang[2,3]

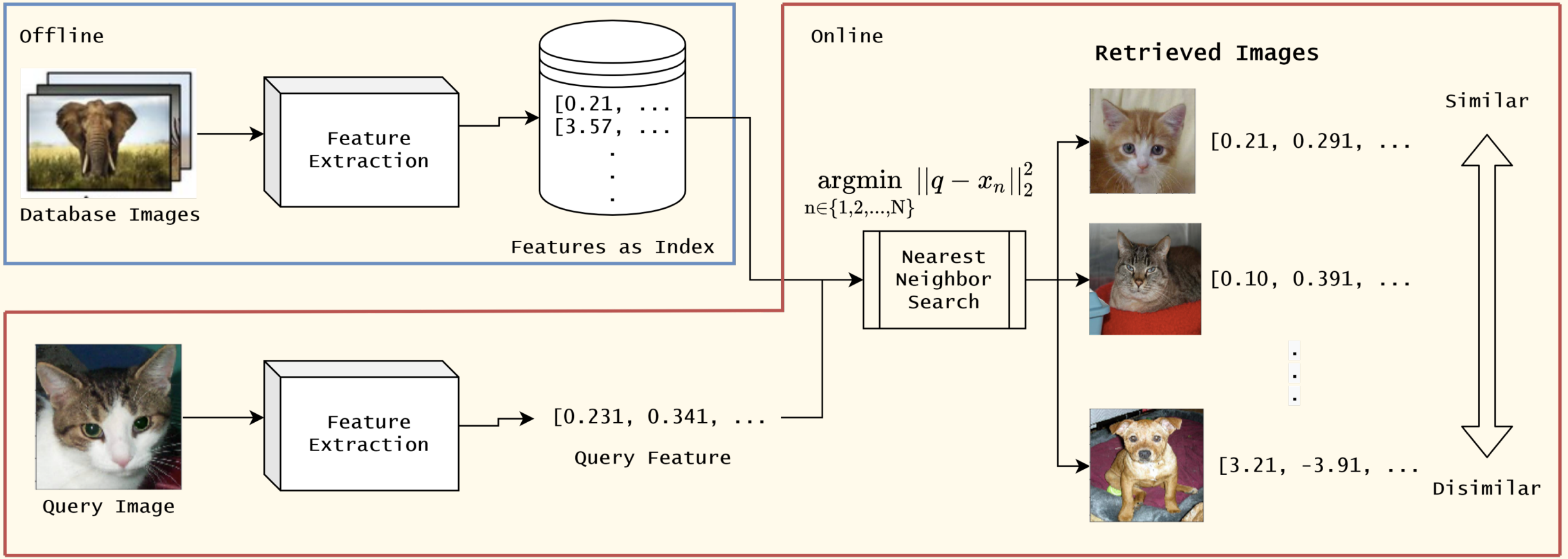
[1]Universiti Malaya, Malaysia
[2]University of Surrey, United Kingdom
[3]iFlyTek-Surrey Joint Research Centre on Artificial Intelligence
[4]Geek+, China

***Equal Contribution**

Table of Content

* ***OrthoHash*** is the proposed method.

Image Retrieval in General

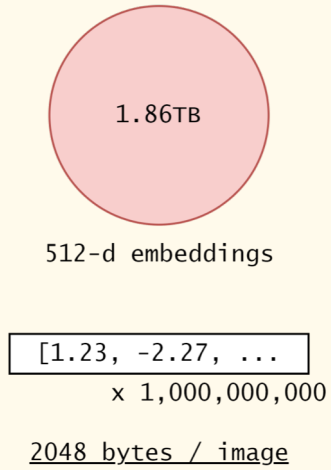


Nearest Neighbor (NN) Search

- Searching: Linear scan, *O*(ND), **slow**

$$\underset{n \in \{1,2,\ldots,N\}}{\operatorname{argmin}} \left|\left|q - x_n\right|\right|_2^2$$

- Storage: **High** memory consumption



Solution: Approximate NN Search

- Aims to improve searching speed
- No need to be exact neighbors
- Methods:
  i. Tree-based search
  ii. Product Quantization
  iii. Binary Hashing

Binary Hashing

- Searching: Low-level operation, **faster**

$$\underset{n\in\{1,2,\ldots,N\}}{\operatorname{argmin}} ||q - x_n||_2^2 \quad \Longrightarrow \quad \texttt{popcount(q XOR } x_n\texttt{)}$$

- Storage: **Lower** memory consumption

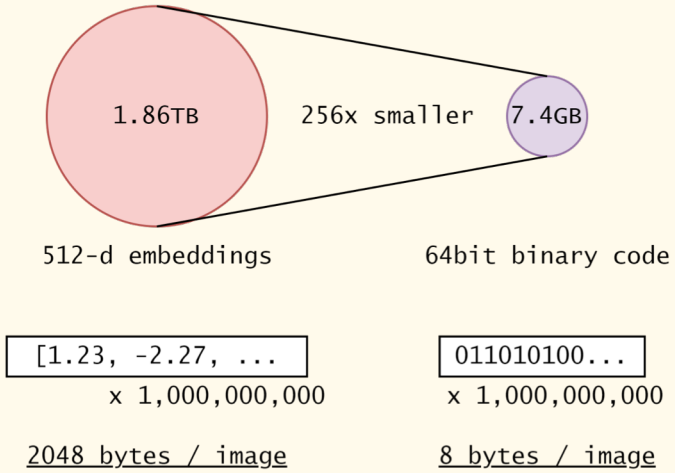
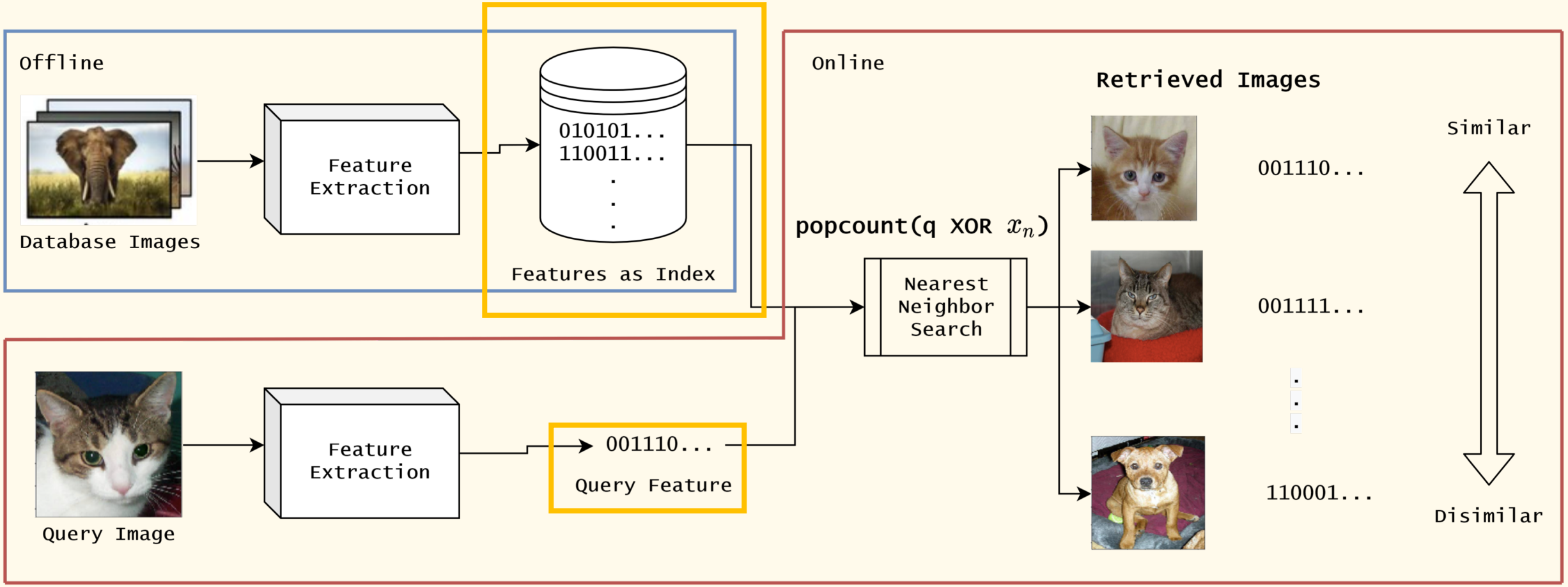
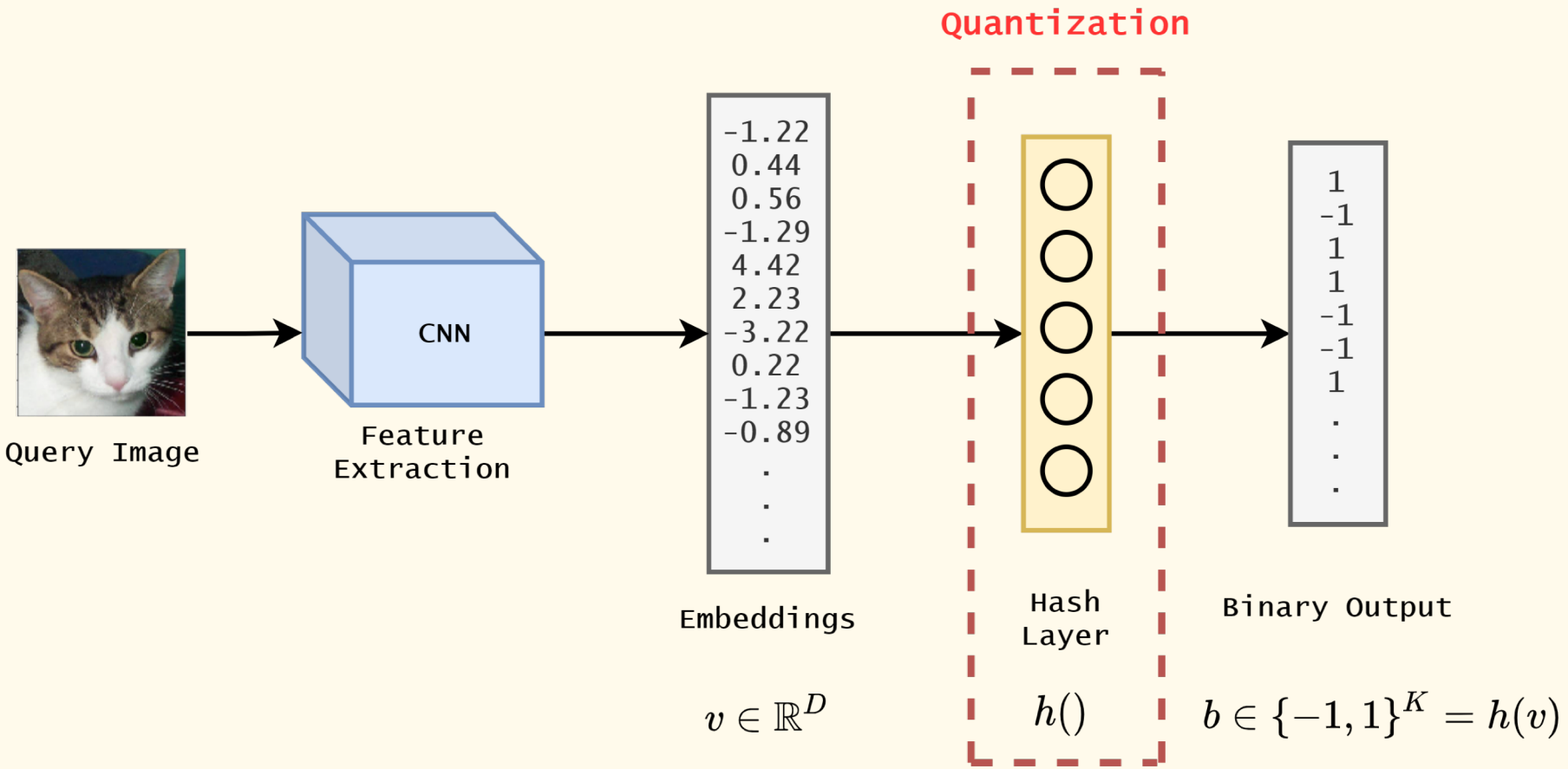


Image Retrieval with Binary Codes



Binary Hashing



Main Objective:

- Learn a hash layer that map embeddings into binary codes

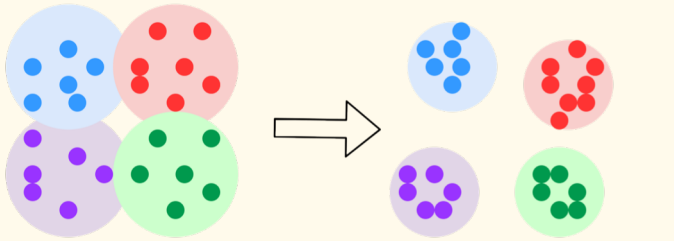
Main Difficulty:

- Sign function is **not differentiable**

Binary Hashing

- More constraints:

1. Discriminative codes



Lower intra-class variance
Higher inter-class distance

2. Minimal quantization error

$$\sum_{i}^{N}\sum_{k}^{K}||f_{ik}-b_{ik}||_2^2$$

3. Maximizing bit capacity – bit balance [1]

For each bit:

$$\sum_{i}^{N} b_{ik} = 0$$

4. Code orthogonality – uncorrelated bit [1]

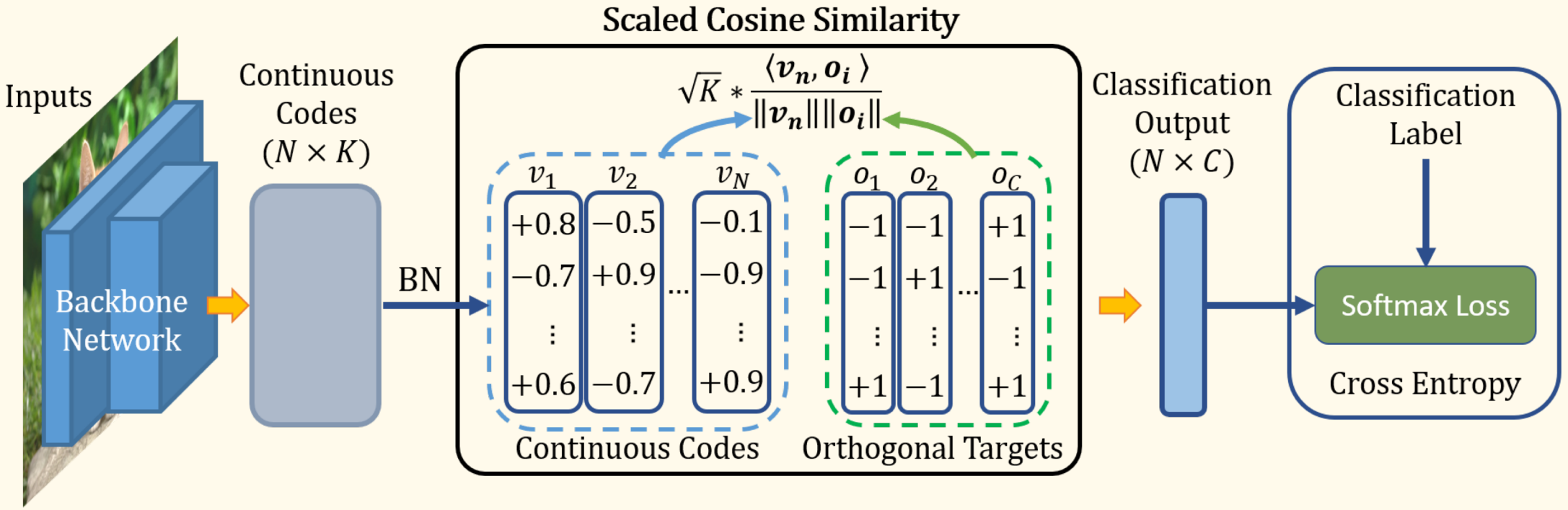
$$B \in \{-1,1\}^{N\times K} \quad , \quad \frac{1}{N}B^T B = I$$

- Too many objectives to learn! Difficult to optimize!

[1] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. NIPS 2009.

Binary Hashing through *OrthoHash*

- How to unify all objectives? – ***OrthoHash***



Binary Hashing through *OrthoHash*

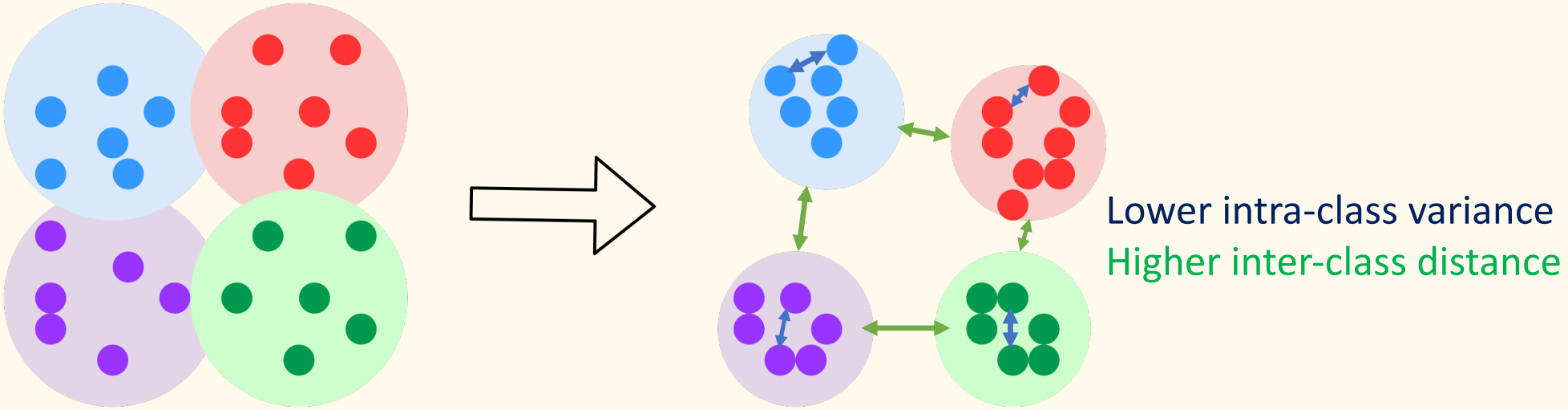
- How to unify all objectives? – ***OrthoHash***
- Properties:
  a) Classification-based learning objective with *cosine/angular margin*
  b) Balanced bits through *Batch Normalization (BN)*
  c) Pre-defined *orthogonal* hash targets
  d) Can learn binary hash codes *end-to-end* without bypassing the non-differentiable sign function (no sign function involved during training)

Binary Hashing through *OrthoHash*

a) Classification-based learning objective with cosine/angular margin

Without Cosine/Angular margin

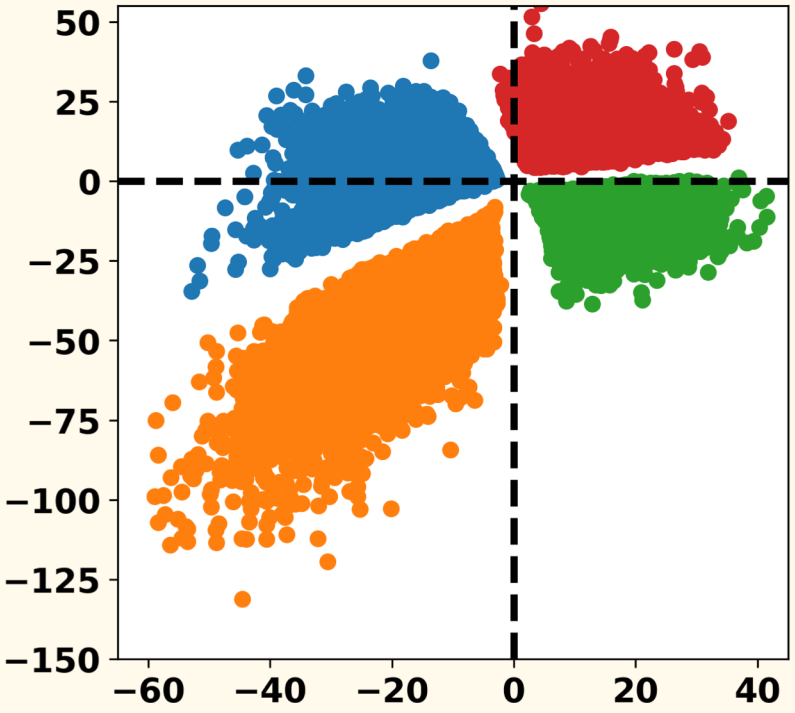
With Cosine/Angular margin



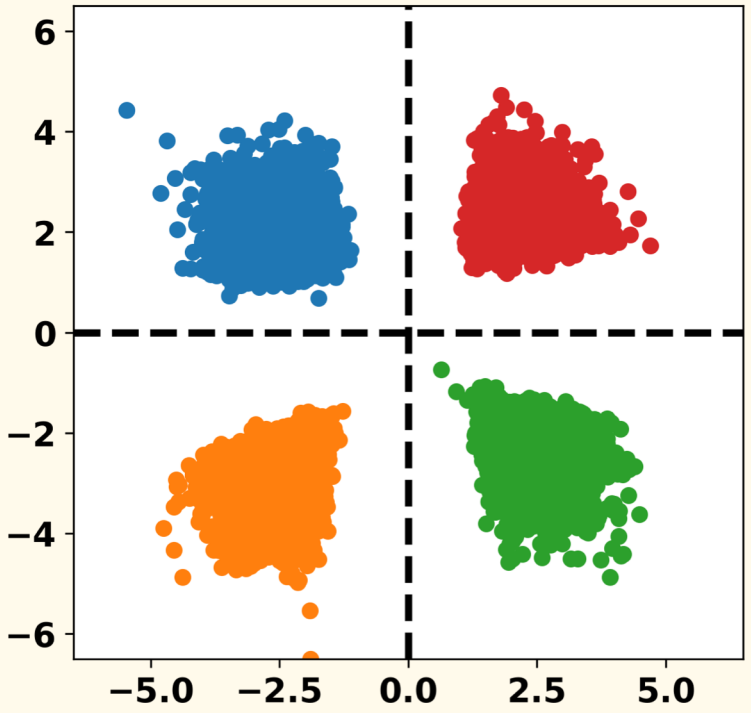
Binary Hashing through *OrthoHash*

b) Balanced bits through Batch Normalization (BN)

ONLY Cross Entropy



Cross Entropy **+ BN**

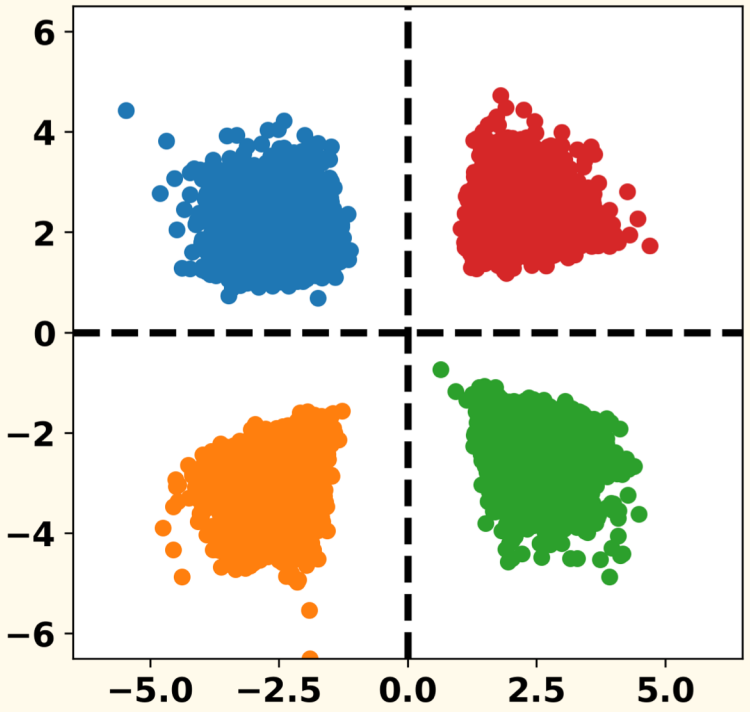


* Cross Entropy represents classification-based objective with *cosine/angular margin*

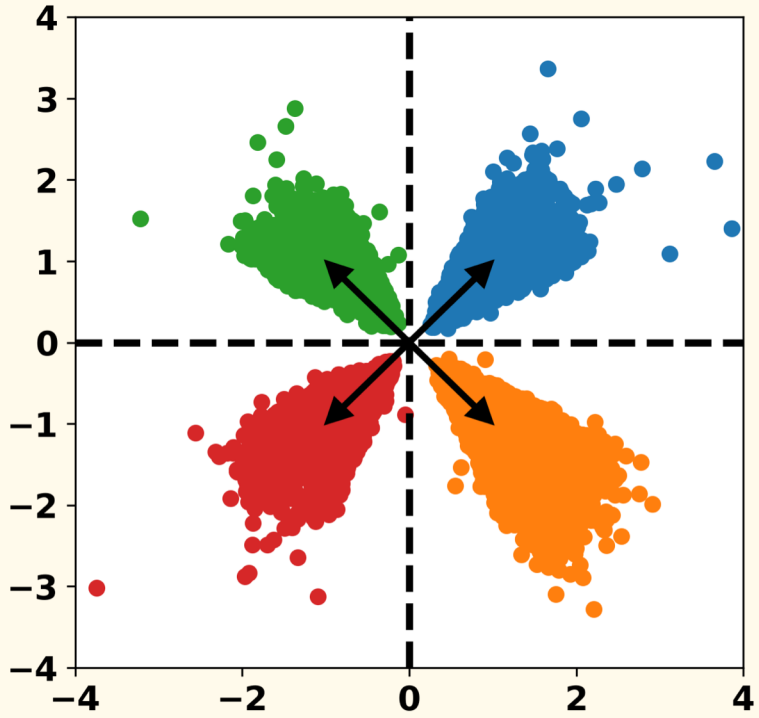
Binary Hashing through *OrthoHash*

c) Pre-defined *orthogonal* hash targets

Cross Entropy + BN

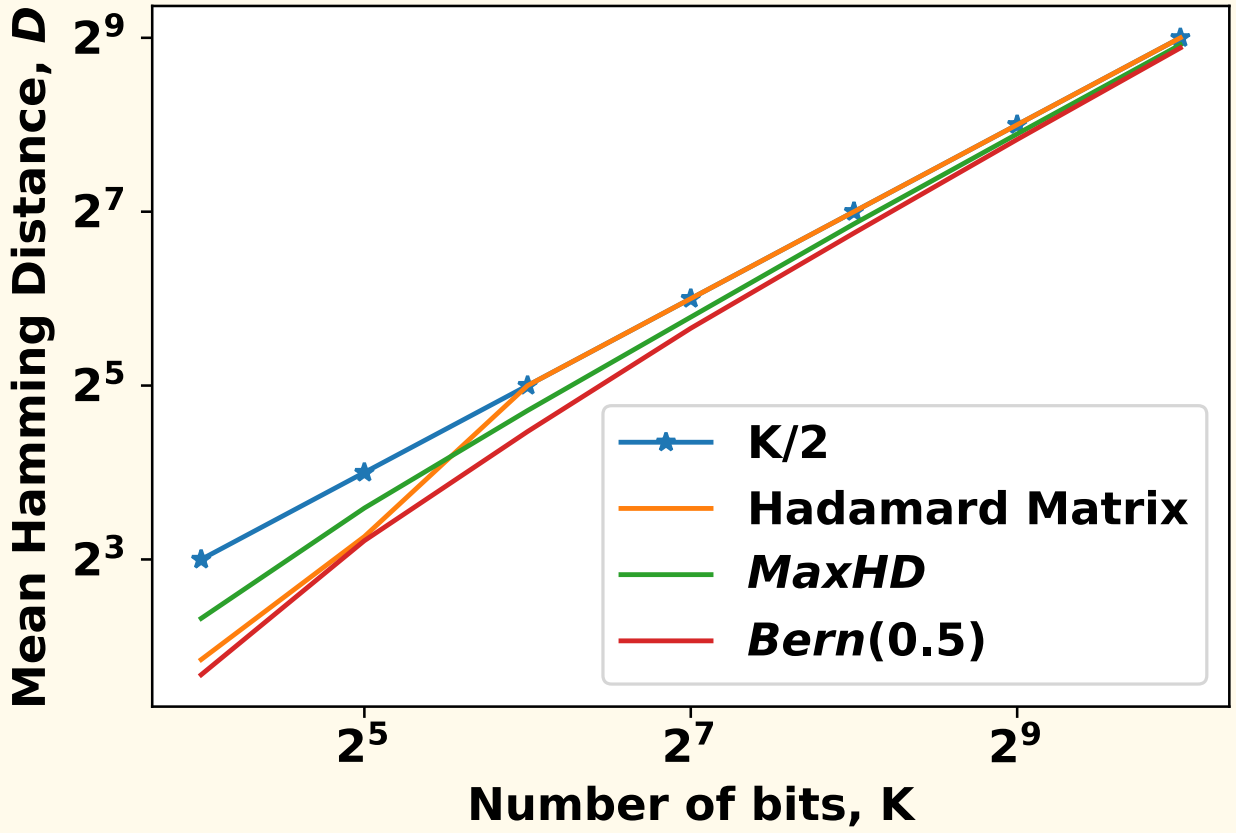


Cross Entropy + BN **+ Orthogonal Target**



Binary Hashing through *OrthoHash*

c) Pre-defined *orthogonal* hash targets



Experiment Results (**Category** Level)

Methods	**ImageNet100** (mAP@1K)				**NUS-WIDE** (mAP@5K)				**MS COCO** (mAP@5K)			
	16	32	64	128	16	32	64	128	16	32	64	128
HashNet[2] [4]	0.343	0.480	0.573	0.612	0.814	0.831	0.842	0.847	0.663	0.693	0.713	0.727
DTSH[3] [44]	0.442	0.528	0.581	0.612	**0.816**	**0.836**	**0.851**	**0.862**	0.699	0.732	0.753	0.770
SDH-C[1] [30]	0.584	0.649	0.664	0.662	0.763	0.792	0.816	0.832	0.671	0.710	0.733	0.742
GreedyHash[1] [40]	0.570	0.639	0.659	0.659	0.771	0.797	0.815	0.832	0.677	0.722	0.740	0.746
JMLH[1] [39]	0.517	0.621	0.662	0.678	0.791	0.825	0.836	0.843	0.689	0.733	0.758	0.768
DPN[1] [11]	0.592	0.670	0.703	0.714	0.783	0.818	0.838	0.842	0.668	0.721	0.752	0.773
CSQ[1] [49]	0.586	0.666	0.693	0.700	0.797	0.824	0.835	0.839	0.693	**0.762**	0.781	0.789
CE[1]	0.350	0.379	0.406	0.445	0.744	0.770	0.796	0.813	0.602	0.639	0.658	0.676
CE+BN[1]	0.533	0.586	0.612	0.617	0.801	0.814	0.823	0.825	0.697	0.721	0.729	0.726
CE+Bihalf[1] [26]	0.541	0.630	0.661	0.662	0.802	0.825	0.836	0.839	0.674	0.728	0.755	0.757
OrthoCos[1]	0.583	0.660	0.702	0.714	0.795	0.826	0.842	0.851	0.690	0.745	0.772	0.784
OrthoCos+Bihalf[1]	0.562	0.656	0.698	0.711	0.804	0.834	0.846	0.852	0.690	0.746	0.775	0.782
OrthoCos+BN[1]	0.606	0.679	**0.711**	**0.717**	0.804	**0.836**	0.850	0.856	**0.709**	**0.762**	**0.787**	**0.797**
OrthoArc+BN[1]	**0.614**	**0.681**	0.709	0.714	0.806	0.833	0.850	0.856	0.708	**0.762**	0.785	0.794

Table 1: Performance of different methods for 4 different bits on different benchmark datasets. All results are run by us. The superscript [1], [2] and [3] indicate point-wise, pair-wise and triplet-wise method respectively. **Bold** values indicate best performance in the column.

Experiment Results (**Instance** Level)

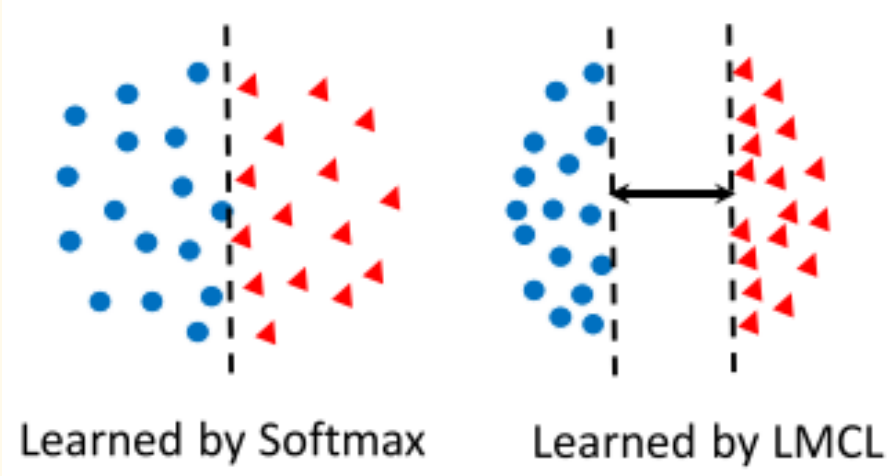
Methods	**GLDv2** (mAP@100)			$\mathcal{R}$**Oxf-Hard** (mAP@all)			$\mathcal{R}$**Paris-Hard** (mAP@all)		
	128	512	2048	128	512	2048	128	512	2048
HashNet[2] [4]	0.018	0.069	0.111	0.034	0.058	0.307	0.133	0.190	0.490
DPN[1] [11]	0.021	0.089	0.133	0.053	0.184	0.303	0.224	0.399	0.562
GreedyHash[1] [40]	0.029	0.108	0.144	0.032	0.251	0.373	0.128	0.531	0.652
CSQ[1] [49]	0.023	0.086	0.114	0.093	0.284	0.398	0.245	0.541	0.649
OrthoCos+BN[1]	**0.035**	**0.111**	**0.147**	**0.184**	**0.359**	**0.447**	**0.416**	**0.608**	**0.669**
R50-DELG-H	-	-	0.125*	-	-	0.471	-	-	0.682
R50-DELG-C	-	-	0.138*	-	-	0.510	-	-	0.715

Table 2: Performance of different methods for 3 different numbers of bits on different instance-level benchmark datasets. All results are run by us. The superscript [1] and [2] indicate point-wise and pair-wise method respectively. **Bold** values indicate best performance in the column. * indicates using 512 $\times$ 512 image inputs, hence different performance as reported by DELG [2]. R50-DELG-H denotes Hamming distance retrieval using the sign of extracted descriptors. R50-DELG-C denotes Cosine distance retrieval using the extracted descriptors.

Why *OrthoHash* performs well?

- Learning **discriminative** codes with pre-defined orthogonal hash targets:
  - Cross entropy with cosine margin [2] **minimize the intra-class distance.**

Low intra-class hamming distance



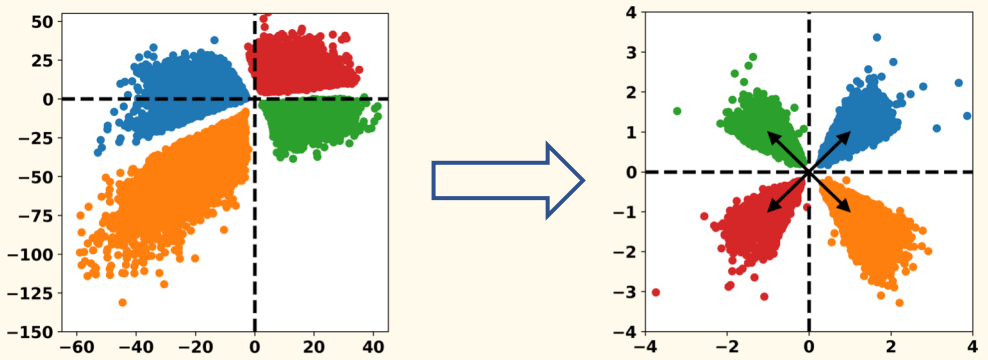
LMCL: Large margin cosine loss

[2] Wang et al. Cosface: Large margin cosine loss for deep face recognition. CVPR 2018.

Why *OrthoHash* performs well?

- Learning **discriminative** codes with pre-defined orthogonal hash targets:
  - Cross entropy with cosine margin [2] **minimize the intra-class distance.**
  - *Orthogonal* targets (like Hadamard matrix) maximize the Hamming distance between every class (**maximize inter-class distance**).

Large inter-class hamming distance

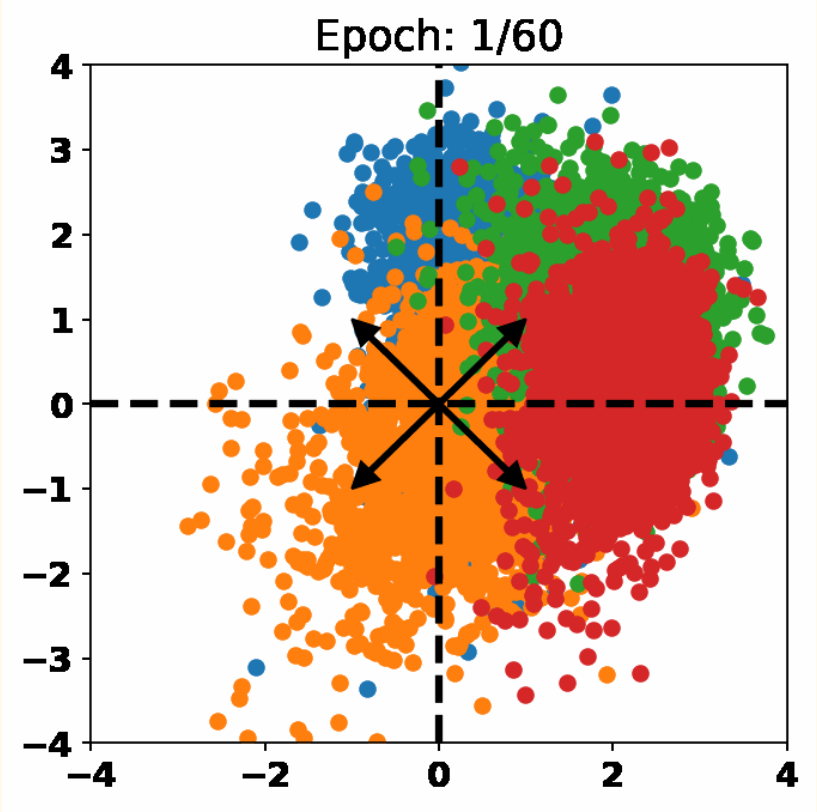


Cross Entropy **OrthoHash**

[2] Wang et al. Cosface: Large margin cosine loss for deep face recognition. CVPR 2018.

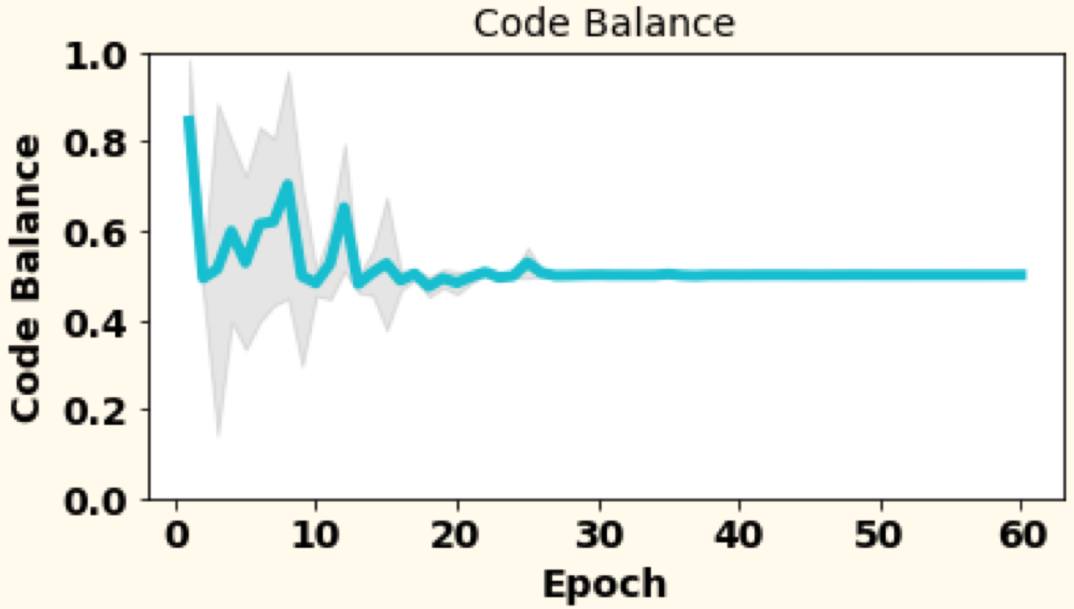
Why *OrthoHash* performs well?

1. Orthogonal hash targets ensure **code orthogonality**.



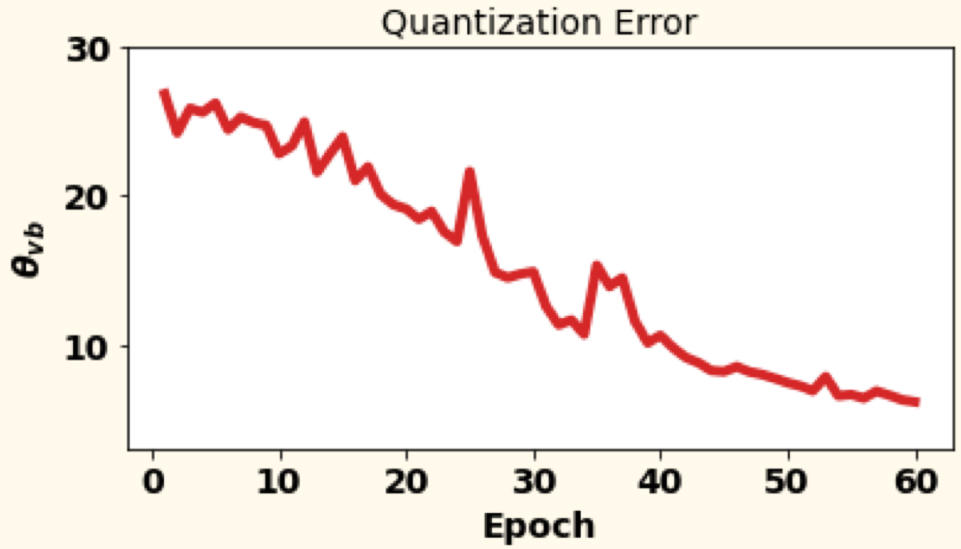
Why *OrthoHash* performs well?

1. Orthogonal hash targets ensure **code orthogonality**.
2. Batch Normalization (BN) layer compute the output with zero-mean, hence, ensure **bit balance**.



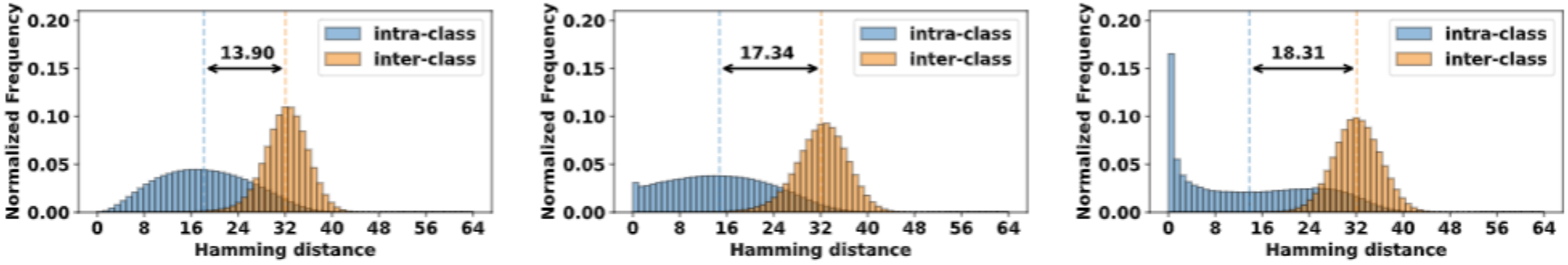
Why *OrthoHash* performs well?

1. Orthogonal hash targets ensure **code orthogonality**.
2. Batch Normalization (BN) layer compute the output with zero-mean, hence, ensure **bit balance**.
3. **Quantization error is minimized** simultaneously.



More analysis on performance

- How the gap between intra-class and inter-class Hamming distances affects the performance?
  - ➢*The larger the gap, the better the performance.*



(a) HashNet [4] (b) GreedyHash [40] (c) OrthoCos+BN

Figure 3: Histogram of intra-class and inter-class Hamming distances with 64-bits ImageNet100. The arrow annotation is the separability in Hamming distances, $\mathbb{E}[D_{inter}] - \mathbb{E}[D_{intra}]$. We normalized the frequency so that sum of all bins equal to 1.

Conclusion

- To achieve good performance in image retrieval with binary hash codes, one must ensure:
  a) Discriminative binary codes
  b) Low quantization errors
  c) Maximize the bit capacity in the codes
  d) Ensure orthogonality in hash centers (average codes for each class)
- We proposed ***OrthoHash***
  - **unify** all the objectives in above (OrthoCos/OrthoArc are variants of OrthoHash)
  - easy to optimize with **only single classification-based** learning objective without bypassing the non-differentiable problem.

NEURAL INFORMATION PROCESSING SYSTEMS 2021

UNIVERSITI MALAYA

UNIVERSITY OF SURREY

Thank you!

https://github.com/kamwoh/orthohash