Complementary Binary Quantization for Joint Multiple Indexing

Qiang Fu[1], Xu Han[1], Xianglong Liu[1*], Jingkuan Song[2], Cheng Deng[3]

[1] *Beihang University, Beijing, China*

[2] *University of Electronic Science and Technology of China, Chengdu, China*

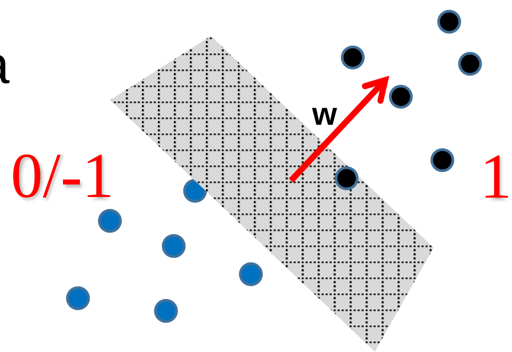
[3] *Xidian University, Xi'an, China*

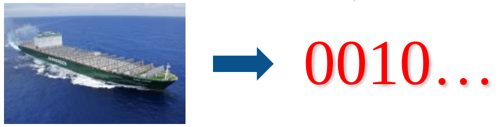
Outline

- **Introduction**
  - Hash-based Nearest Neighbor Search (NNS) Solution
  - Multi-table Indexing
- **Complementary Binary Quantization (CBQ)**
  - Complementary Multi-Table Quantization Formulation
  - Joint table learning
  - Algorithm details
- **Experiments**
  - Euclidean Nearest Neighbor Search
  - Semantic Nearest Neighbor Search
- **Reference**

Introduction: Hash-based NNS Solution

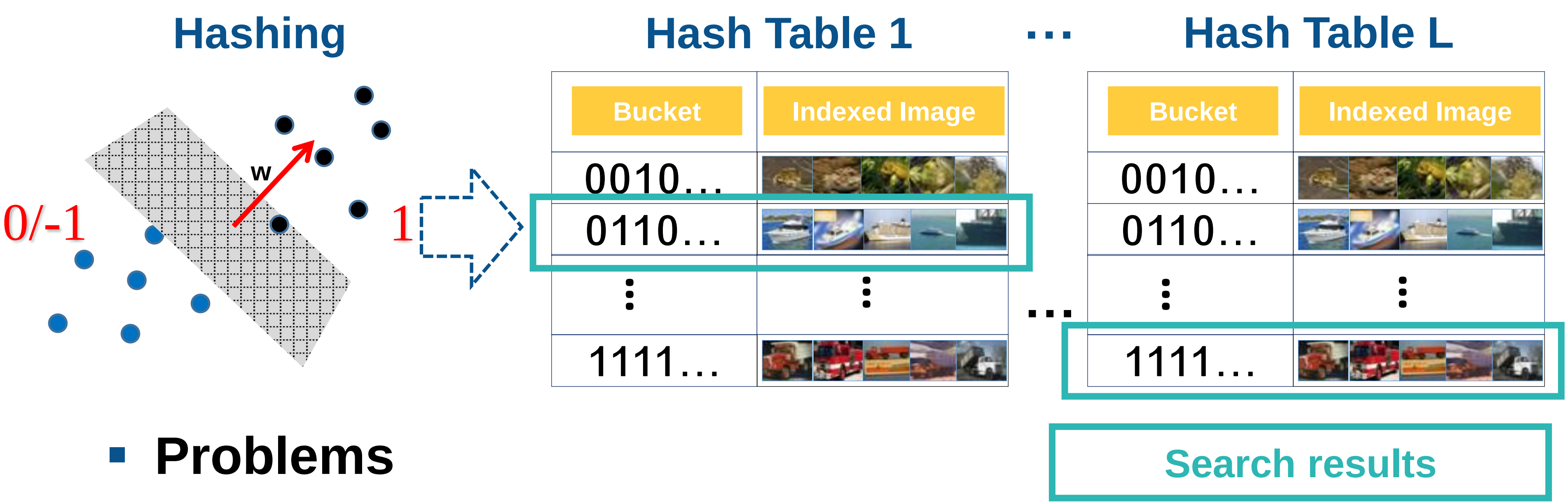
- **Hash-based solution**
  - Encode data to binary codes
  - Compressed storage and efficient computation
  - Widely-used in many applications like image search, feature match...
- **Locality Sensitive Hashing (LSH)**
  - Close points in the original space have similar hash codes
- **Projection based hashing**
  - leverage the information contained in the data
  - ITQ, SH, AGH…
- **Prototype based hashing**
  - Characterize the natural data relations better
  - SPH, ABQ…





Introduction: Multi-table Indexing

- **Multi-table indexing**
  - Build multiple hash tables and probe multiple buckets to improve the search performance
  - Complementary multi-table method
    - maximally cover the nearest neighbors using fewer tables



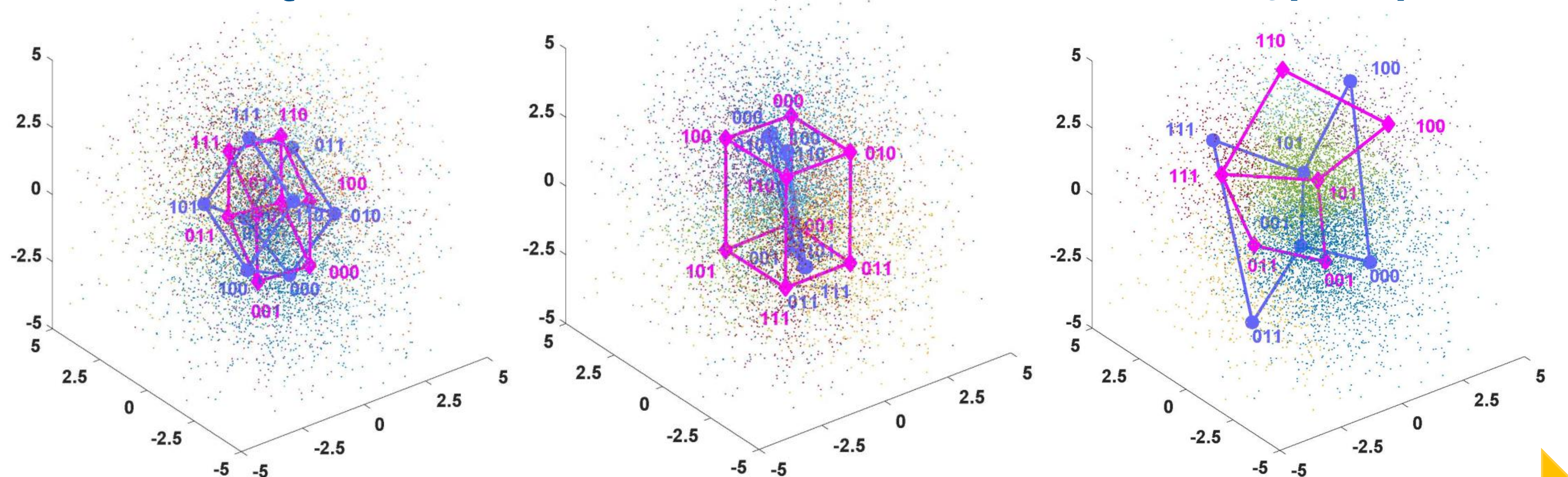
- **Problems**
  - Suffer from the table redundancy still
  - Describe the data distribution and relation not well

Complementary Binary Quantization

- **Goal and Motivation**
  - **Complementary:** jointly learn the multiple hash tables
  - **Informative**: use prototype based hashing quantization
- **Formulation**

$$\min_{\{P^{(l)}\},\{C^{(l)}\},\lambda} L = L_{quan} + \mu L_{align}$$

ITQ CH CBQ(ours)



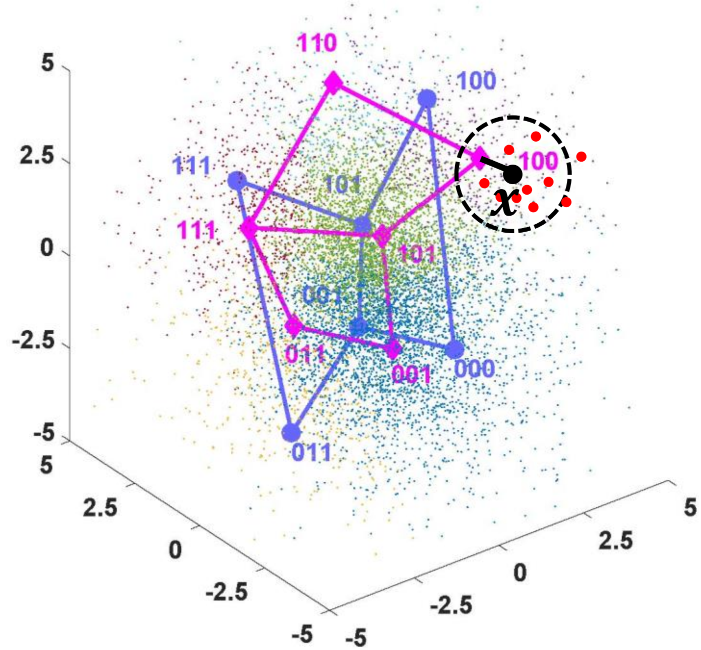
capture the data distribution better, improve the table complementarity more

CBQ: Formulation

- **Multi-Table Quantization Loss**
  - Consider the nature of the multi-index search, only at least one of its nearest prototypes in different tables needed for the correct search results
  - Select the nearest prototype from all tables

$$L_{quan} = \frac{1}{n}\sum_{i=1}^{n} d_o^2\left(x_i, p_{k^*}^{(l^*)}\right)$$

$$d_o\left(x_i, p_{k^*}^{(l^*)}\right) \le d_o\left(x_i, p_k^{(l)}\right), (l, k) \neq (l^*, k^*)$$



$y = 100$

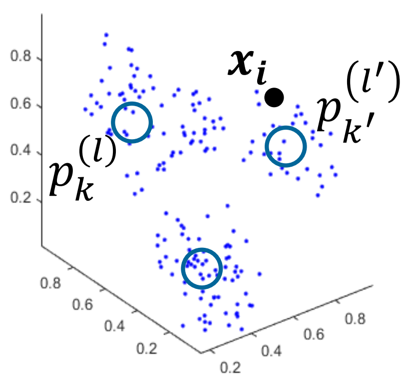
CBQ: Formulation

- **Space Alignment Loss**
  - Concentrate on the distance consistence so that codes in Hamming space will be aligned with the original data distribution

$$L_{align} = \frac{1}{nM} \sum_{i=1}^{n} \sum_{l=1}^{L} \sum_{k=1}^{|P^{(l)}|} \left\| \lambda d_o\left(x_i, p_k^{(l)}\right) - d_h\left(c_{k^*}^{(l^*)}, c_k^{(l)}\right) \right\|^2$$

    - $P^{(l)}$ is the number of prototypes for the l-th table
    - $M = \sum_{l=1}^{L} |P^{(l)}|$

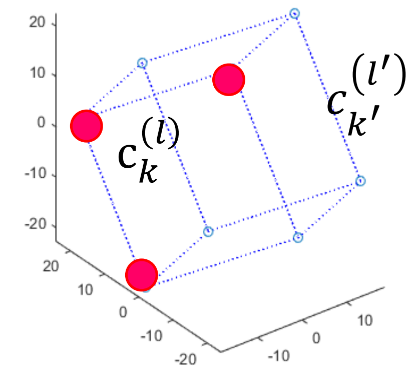
Using a small subset of binary codes



the original space

$$d_h\left(c_{k'}^{(l')}, c_k^{(l)}\right) \approx \lambda d_o\left(p_{k'}^{(l')}, p_k^{(l)}\right)$$

$$d_o\left(x_i, p_k^{(l)}\right) \approx d_o\left(p_{k^*}^{(l^*)}, p_k^{(l)}\right)$$



the Hamming space

CBQ: Joint Table Learning

- **Reformulation for Joint Table Learning**
  - construct a one-to-one mapping that converts the original prototype index (l, k) to a uniform one $m \in \{1, 2, \dots, M\}$

$$\min_{P,C,\lambda} L = \frac{1}{n}\sum_{i=1}^{n} d_o^2\left(x_i, p_{m_i^*}\right) + \frac{\mu}{nM}\sum_{i=1}^{n}\sum_{m=1}^{M}\left\|\lambda d_o(x_i, p_m) - d_h\left(c_{m_i^*}, c_m\right)\right\|^2$$

$$s.t. \quad c_m \in \{-1, 1\}^b; \pi(c_m) \le L, d_o\left(x_i, p_{m_i^*}\right) \le d_o(x_i, p_m), m \ne m_i^*$$

- **Alternating Optimization**
  - **(1) Incomplete Coding** (optimize C)
    - Find a sub-codebook most consistent with the prototypes
  - **(2) Prototype Pursuit** (optimize P)
    - Find a prototype set that will shrink
  - **(3) Rescaling** (optimize λ)
    - Find a proper space scaling

Algorithm 1 Complementary Binary Quantization (CBQ).

Input: Training set $\mathbf{X}$, hash table number L, code length b per table.

Output: Hash functions $\{h^{(l)}\}_{l=1}^{L}$

```
Initialize prototype set 𝒫 and the assignment index m_i^* for X using K-means.
Initialize the scale variable λ according to (14).
repeat
    for m' = 1, 2, ..., M do
        Update the code set 𝒞 using the local optimal binary code c_{m'} for p_{m'} by solving (8).
    end for
    Update 𝒫 by iteratively solving (10) and (11).
    Update λ according to (13).
until convergence
Assign 𝒫 and 𝒞 to L hash tables, generating {h^(l)}_{l=1}^L.
```

CBQ: Algorithm Details

- **Table Assignment**
  - Use a random assignment strategy
    - no identical hash codes in the same hash table
    - the prototype numbers of each hash table should be balanced
- **Initialization**
  - P: classical k-means clustering
  - λ: $M(L \times 2^b)$ prototypes and codes
- **Generate long hash codes ($L \times 2^{b'}$)**
  - Use product quantization method
- **Complexity**
  - Training: $O\left(\left(L \times 2^b\right)^2 nd\right) = O(4^b L^2 nd)$, almost linear to n
  - Testing: $O(2^b Ld)$, close to constant, almost same to LSH and ITQ

Experiments

- **Datasets**
  - Euclidean Nearest Neighbor Search (NNS)
    - **SIFT-1M**: 1M 128-D SIFT, **GIST-1M**: 1M 960-D GIST
  - Semantic Nearest Neighbor Search (NNS)
    - **CIFAR-10:** 60K 384-D GIST**, NUS-WIDE:** 269K 4096-D Conv feature
- **Baselines**
  - State-of-the-art unsupervised hashing
    - Projection-based: **LSH**, **ITQ**, **SH**, **AGH**
    - Prototype-based: **SPH** , **ABQ**
  - Multi-table hashing methods: **CH**, **BCH**
- **Settings**
  - 10,000 training samples and 1,000 queries on each set
  - Hash code length: $B = 24, b = 3$(each subspace)

Experiments: Euclidean NNS

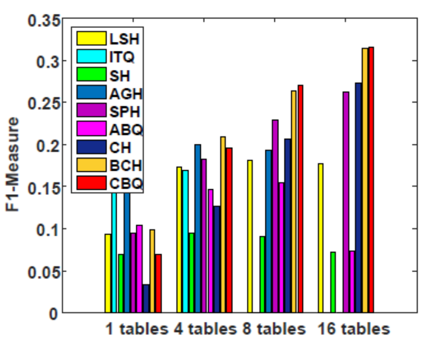
- **Groundtruth**
  - the top 5‰ points with the smallest Euclidean distances
- **Hamming distance ranking**

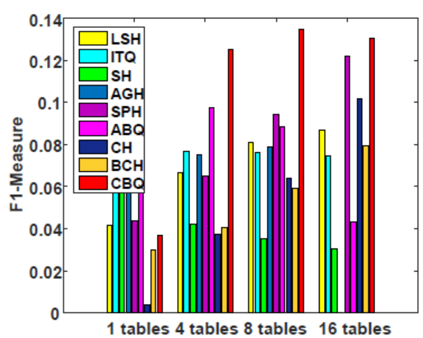
$$d(\mathbf{x}_q, \mathbf{x}_i) = \min_{l=1,\ldots,L} d_h(\mathbf{y}_q^{(l)}, \mathbf{y}_i^{(l)}).$$

Method	SIFT-1M					GIST-1M				
	L=1	L=4	L=8	L=16	Train Time	L=1	L=4	L=8	L=16	Train Time
LSH	27.71 ±0.63	29.40 ±0.81	29.08 ±2.47	29.62 ±0.79	0.04	9.67 ±0.61	10.54 ±0.42	11.54 ±0.24	11.76 ±0.84	5.11
ITQ	41.06 ±1.53	30.70 ±0.33	-	-	3.40	26.83 ±0.28	16.62 ±0.60	14.20 ±0.31	12.59 ±0.80	12.83
SH	48.81 ±0.84	19.64 ±3.32	14.55 ±2.59	10.53 ±0.87	0.19	13.72 ±1.10	8.05 ±1.13	5.90 ±0.57	4.97 ±0.44	2.22
AGH	31.55 ±1.71	30.01 ±1.91	28.07 ±1.79	-	0.40	12.55 ±1.25	11.28 ±0.50	11.62 ±0.50	-	4.51
SPH	39.41 ±0.89	41.49 ±0.59	41.78 ±0.98	41.61 ±0.38	5.30	22.76 ±0.59	19.68 ±0.52	19.50 ±0.44	19.42 ±0.55	18.39
ABQ	51.53 ±1.30	32.88 ±0.57	24.61 ±0.69	10.60 ±0.61	36.08	**29.28** ±0.60	17.44 ±0.55	14.22 ±0.67	4.56 ±0.06	74.00
CH	50.51 ±0.94	52.28 ±0.35	53.05 ±0.80	54.12 ±0.82	0.24	18.74 ±0.78	23.83 ±0.41	24.74 ±0.41	25.68 ±0.53	2.94
BCH	45.81 ±0.93	53.30 ±0.44	55.69 ±0.60	57.37 ±0.31	258.33	14.02 ±0.65	17.04 ±0.40	18.59 ±0.60	19.94 ±0.71	332.61
CBQ(ours)	**52.39** ±0.71	**55.95** ±0.68	**57.38** ±0.51	**57.76** ±0.79	18.75	26.55 ±1.09	**28.87** ±0.72	**29.28** ±1.18	**29.38** ±0.83	27.51

Table 1: The AP @100 (%) and time cost (seconds) of different hashing methods on SIFT-1M and GIST-1M.

- **Hash table lookup** (Hamming radius ≤3)





Method		SIFT-1M		GIST-1M	
		F1 Measure	Time Cost	F1 Measure	Time Cost
PQ		23.18	1.22	13.12	3.39
CBQ	L=8	19.52	0.02	13.45	0.14
	L=16	27.08	0.04	13.07	0.26

< 1.SIFT-1M 2.GIST-1M

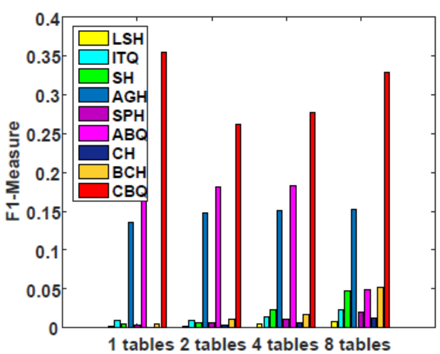
Experiments: Semantic NNS

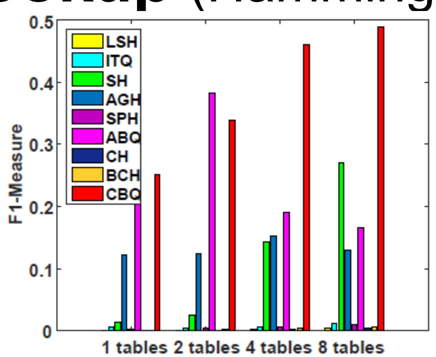
- **Groundtruth**
  - those samples with common tags as the query
- **Hamming distance ranking** $d(\mathbf{x}_q, \mathbf{x}_i) = \min_{l=1,\dots,L} d_h(\mathbf{y}_q^{(l)}, \mathbf{y}_i^{(l)}).$

METHOD	CIFAR-10					NUS-WIDE				
	L=1	L=2	L=4	L=8	TRAIN TIME	L=1	L=2	L=4	L=8	TRAIN TIME
LSH	17.58 ±0.76	18.40 ±1.10	19.85 ±0.65	20.74 ±0.70	1.17	37.96 ±0.50	40.61 ±1.76	41.34 ±1.64	42.22 ±1.22	1.13
ITQ	31.01 ±0.63	27.37 ±0.84	26.36 ±0.60	26.62 ±0.32	11.65	49.19 ±0.30	47.38 ±0.59	45.84 ±0.73	46.37 ±0.84	8.17
SH	18.22 ±0.87	14.84 ±0.91	13.03 ±0.18	12.65 ±0.25	6.86	38.43 ±1.32	37.41 ±0.69	35.22 ±0.68	36.39 ±0.57	6.32
AGH	32.23 ±1.39	31.11 ±2.51	29.14 ±2.42	29.44 ±1.24	1.93	49.62 ±2.66	49.22 ±1.80	49.70 ±1.00	48.53 ±1.79	1.69
SPH	22.64 ±1.38	22.17 ±0.29	22.03 ±0.44	22.55 ±0.13	33.95	43.21 ±1.22	43.71 ±1.07	43.67 ±0.66	44.08 ±0.82	30.13
ABQ	18.94 ±5.26	10.62 ±0.88	10.77 ±0.67	10.97 ±0.46	32.67	38.73 ±3.75	36.98 ±2.92	36.44 ±1.81	34.41 ±1.52	36.42
CH	18.48 ±0.27	22.02 ±0.82	24.75 ±1.34	26.11 ±0.36	8.12	38.46 ±0.59	42.32 ±0.56	44.52 ±0.71	45.72 ±0.83	6.90
BCH	18.52 ±1.10	19.95 ±1.06	19.22 ±1.69	22.44 ±1.18	747.89	37.95 ±2.54	39.46 ±0.86	38.70 ±1.48	39.07 ±1.96	903.97
CBQ(OURS)	**39.66** ±1.79	**39.37** ±1.89	**36.63** ±1.51	**36.62** ±1.22	59.60	**51.92** ±1.53	**54.08** ±1.86	**52.18** ±0.91	**51.14** ±1.76	53.69

Table 2: MAP (%) and time cost (seconds) of different hashing methods on CIFAR-10 and NUS-WIDE.

- **Hash table lookup** (Hamming radius ≤3)





< 1.CIFAR-10 2.NUS-WIDE

Reference

1. Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In ACM STOC , pages 604–613, 1998.
2. Yunchao Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In IEEE CVPR , pages 817–824, 2011.
3. Jae-Pil Heo, Youngwoon Lee, Junfeng He, Shih-Fu Chang, and Sung-Eui Yoon. Spherical hashing. In IEEE CVPR , pages 2957–2964, 2012.
4. Zhujin Li, Xianglong Liu, Junjie Wu, and Hao Su. Adaptive binary quantization for fast nearest neighbor search. In ECAI , pages 64–72, 2016.
5. Hao Xu, Jingdong Wang, Zhu Li, Gang Zeng, Shipeng Li, and Nenghai Yu. Complementary hashing for approximate nearest neighbor search. In IEEE ICCV , pages 1631–1638, 2011.
6. Xianglong Liu, Cheng Deng, Yadong Mu, and Zhujin Li. Boosting complementary hash tables for fast nearest neighbor search. In AAAI , pages 4183–4189, 2017.
7. Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. IEEE TPAMI , 33(1):117–128, January 2011.
8. Yan Xia, Kaiming He, Fang Wen, and Jian Sun. Joint inverted indexing. In IEEE ICCV , pages 3416–3423, 2013.
9. Mohammad Norouzi, Ali Punjani, and David J. Fleet. Fast search in hamming space with multi-index hashing. In IEEE CVPR , pages 3108–3115, 2012.

Thank you!

IJCAI 2018