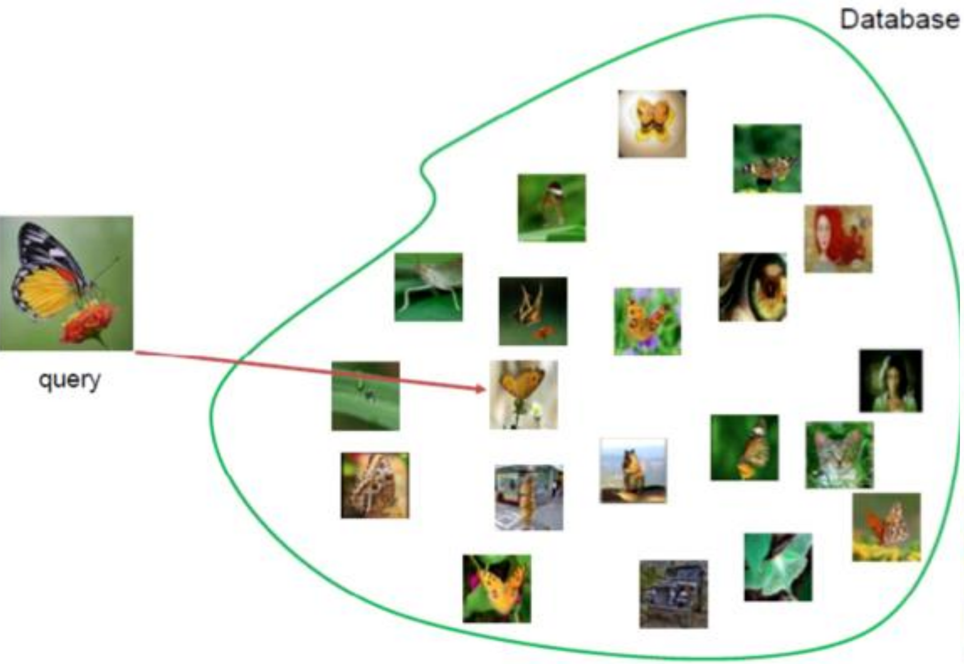
Supervised Hashing for Image Retrieval via Image Representation Learning

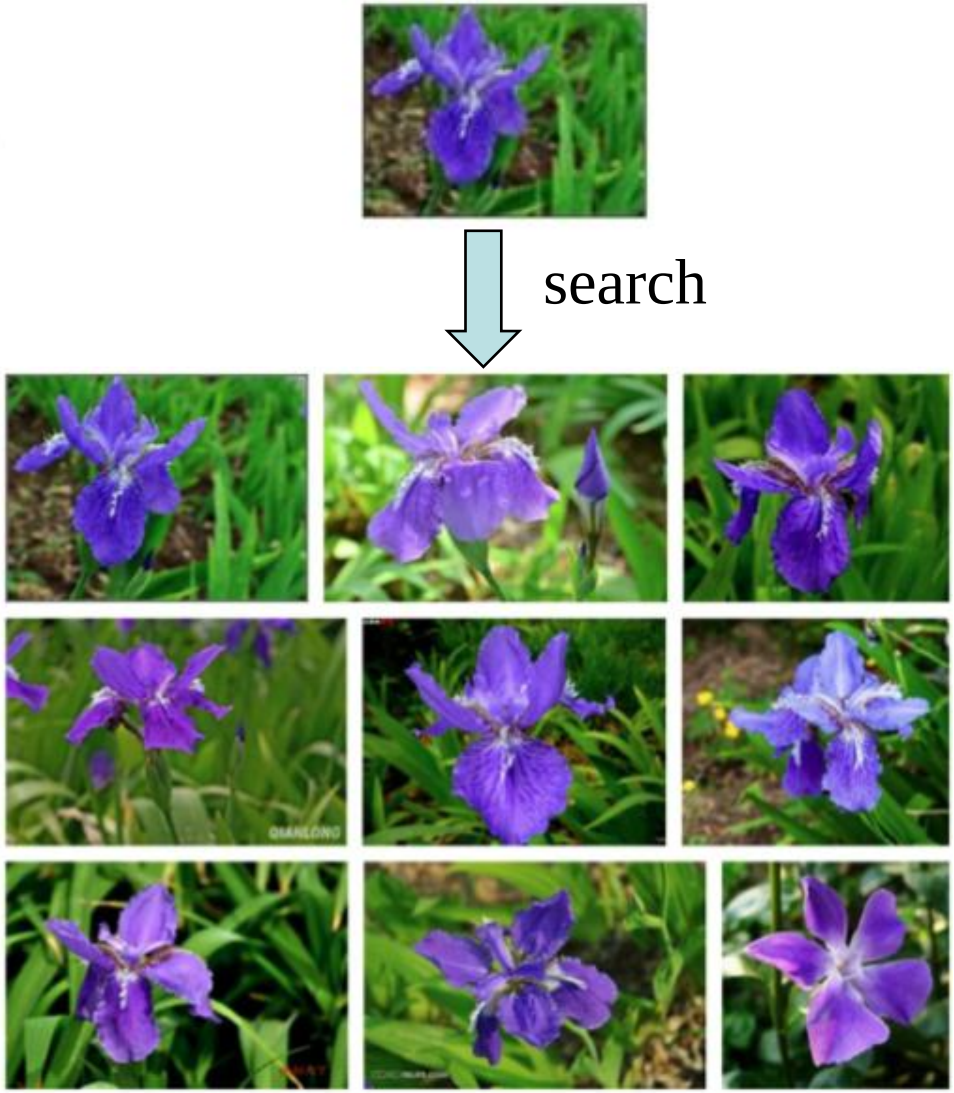
Rongkai Xia, Yan Pan, Cong Liu (**Sun Yat-Sen University**)
Hanjiang Lai, Shuicheng Yan (**National University of Singapore**)

Finding Similar Images

Task: given a query image, find its nearest neighbors in an image database.



Baidu识图



Challenges

Query image

features

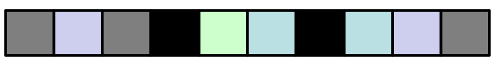
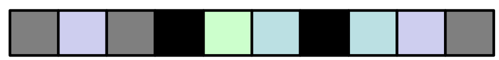
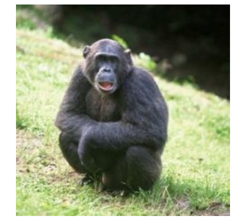


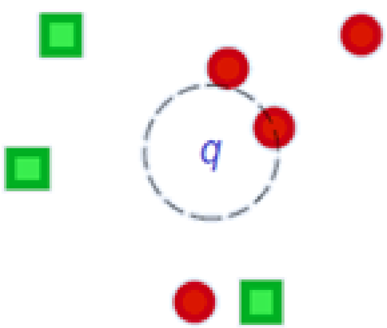
Image database

features





features

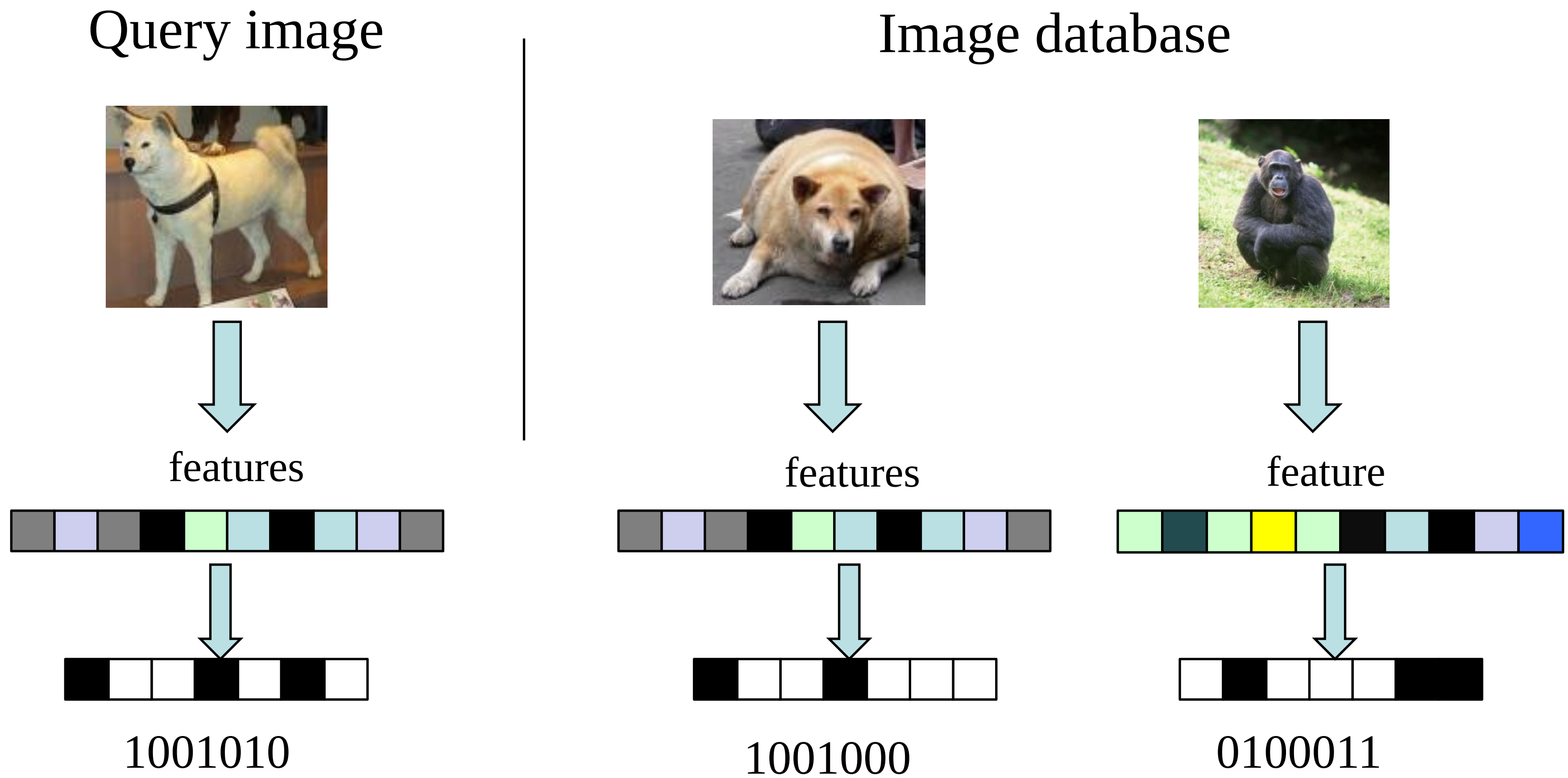


Nearest neighbors search

Challenge： how to efficiently search over millions or billions of data?

- e.g., if each image is represented by a 512-dim GIST vector, one needs 20G memory to store 10 million images.

Similarity-Preserving Hashing

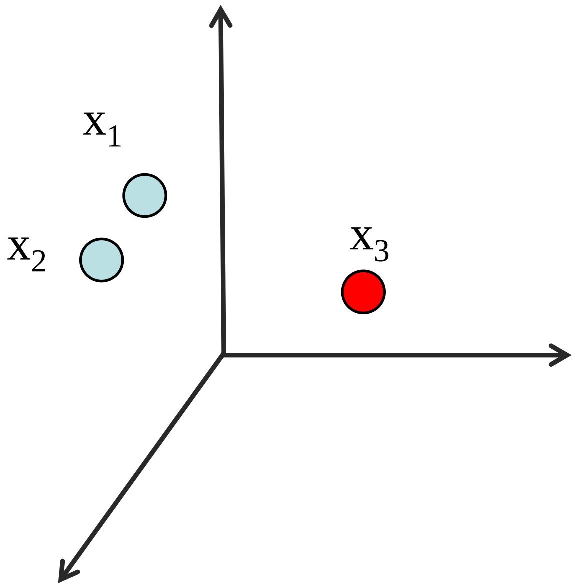


Images are represented by **binary codes**.

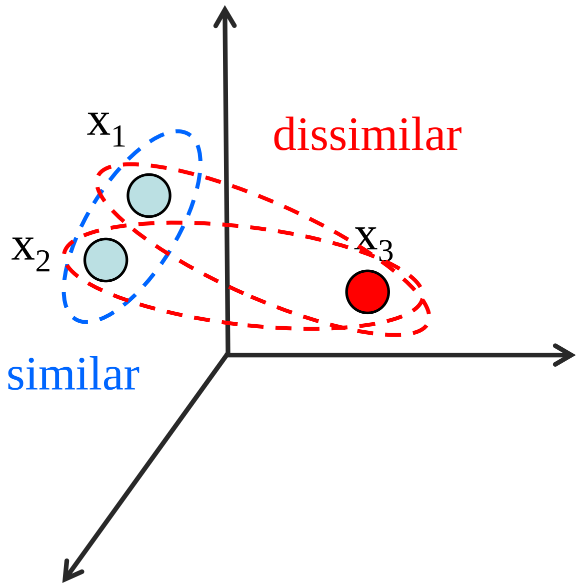
- Efficient retrieval via bitwise operations.
- Space-efficient storage
  - e.g., 10 million images in 32M memory, providing each image in 128-bit code

Key question: how to preserve similarities?

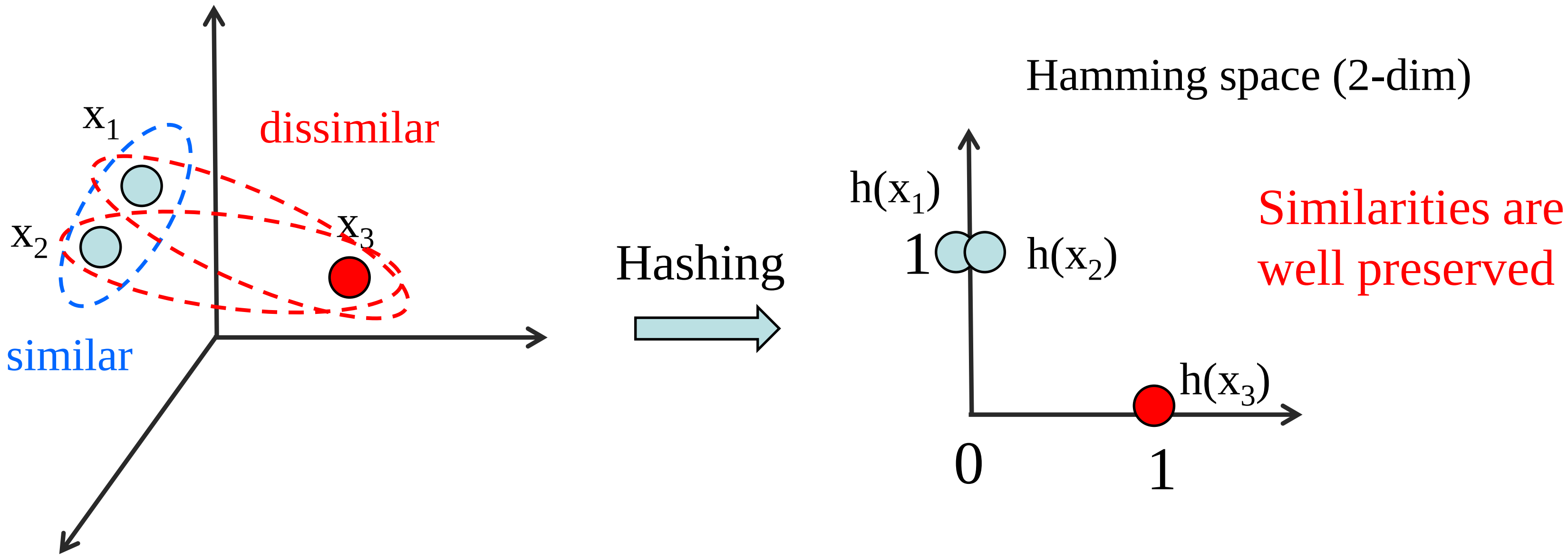
Similarity-Preserving Hashing



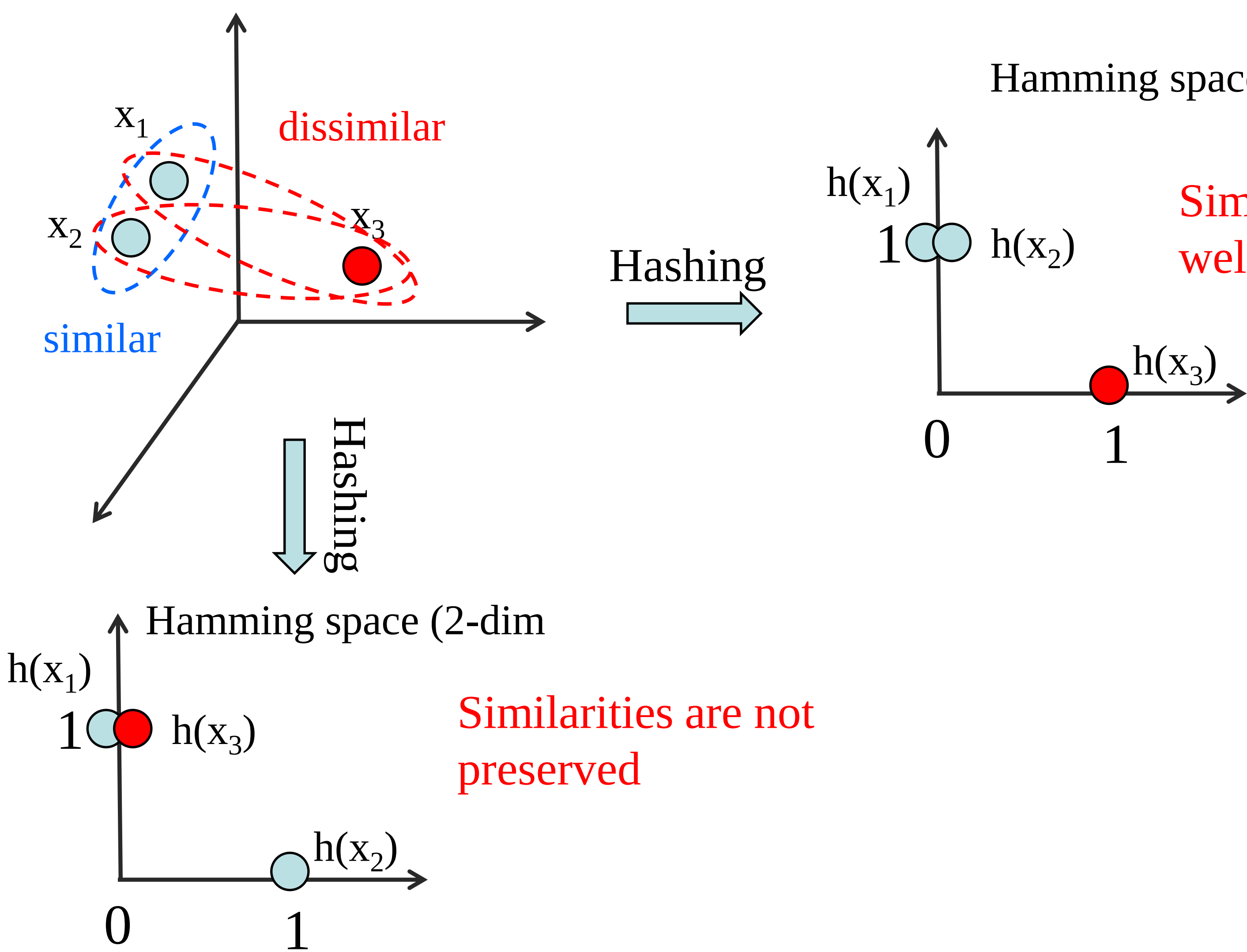
Similarity-Preserving Hashing



Similarity-Preserving Hashing



Similarity-Preserving Hashing



Related Work

Long codes are needed to preserve similarities.

Unsupervised Hashing

LSH[Gionis et al. VLDB,1999]
KLSH[Kulis and Grauman. PAMI,2012]
SH[Weiss and Torralba. NIPS,2008]
ITQ[Gong and Lazebnik. CVPR,2011]

Learn compact codes by using label information.

Semi-supervised Hashing

SSH[Wang et al. CVPR,2010]

Supervised Hashing

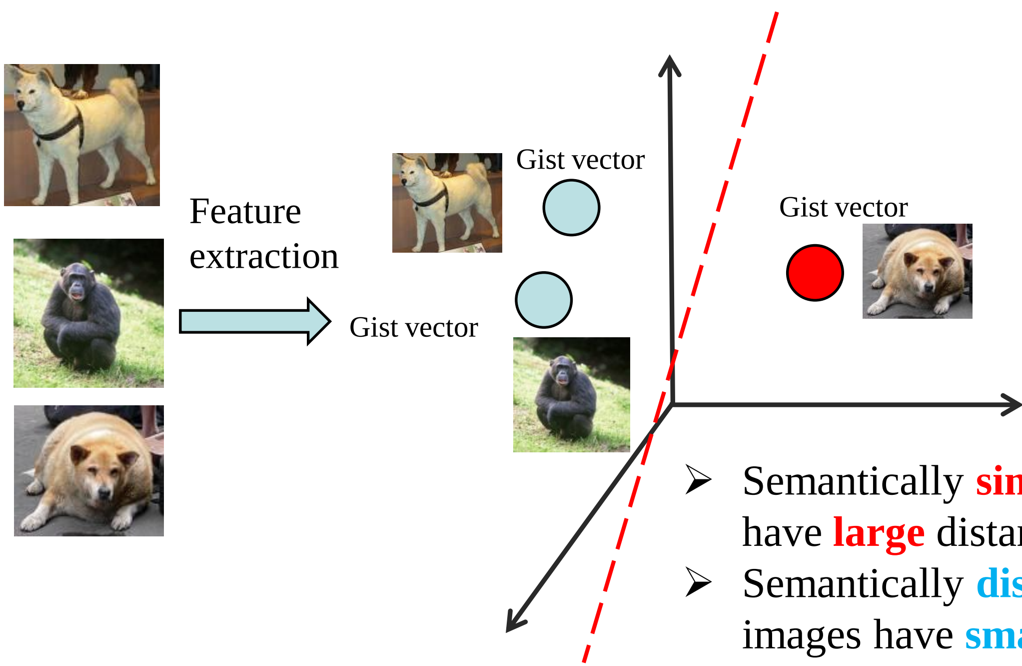
MLH[Norouzi and Blei. ICML,2011]
BRE[Kulis and Darrell. NIPS,2009]
KSH[Liu et al. CVPR,2012]
TSH[Lin et al. ICCV,2013]

Motivation

- In most existing methods, each image is firstly encoded by a vector of some hand-crafted visual descriptor (e.g., GIST, BoW, SIFT)
- **Concern:** the chosen hand-crafted visual features do not necessarily guarantee to accurately preserve the semantic similarities of image pairs.
  - e.g., a pair of semantically similar/dissimilar images may not have feature vectors with relatively small/large Euclidean distance.

Motivation

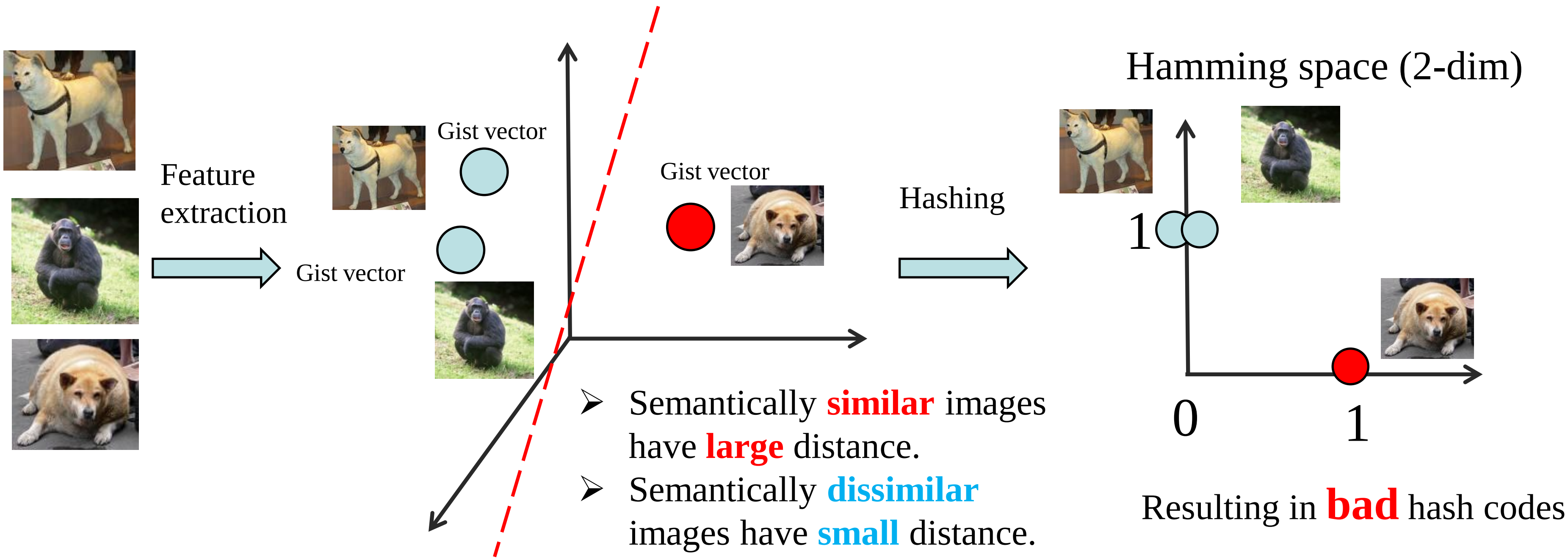
- In most existing methods, each image is firstly encoded by a vector of some hand-crafted visual descriptor (e.g., GIST, BoW, SIFT) .
- **Concern**: the chosen hand-crafted visual features do not necessarily guarantee to accurately preserve the semantic similarities of image pairs.
  - e.g., a pair of semantically similar/dissimilar images may not have feature vectors with relatively small/large Euclidean distance.



- Semantically **similar** images have **large** distance.
- Semantically **dissimilar** images have **small** distance.

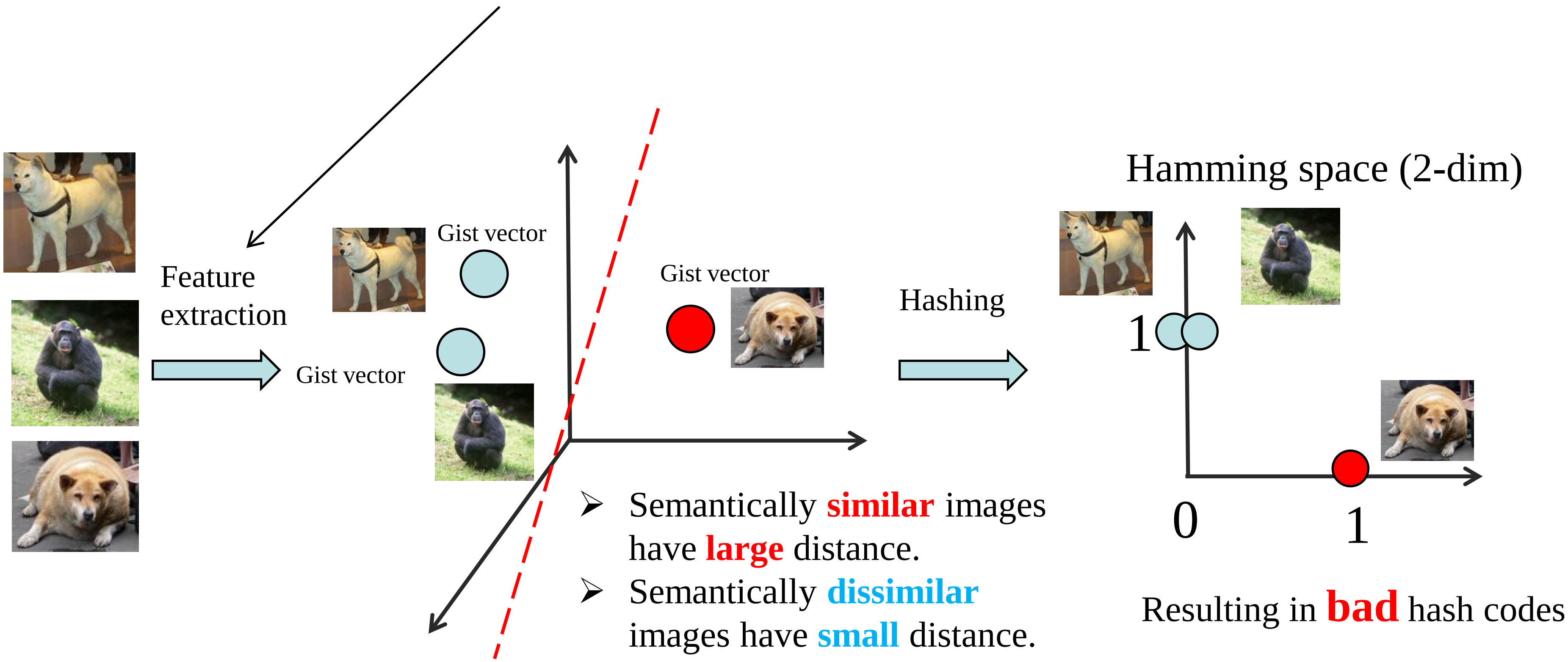
Motivation

- In most existing methods, each image is firstly encoded by a vector of some hand-crafted visual descriptor (e.g., GIST, BoW, SIFT)
- **Concern:** the chosen hand-crafted visual features do not necessarily guarantee to accurately preserve the semantic similarities of image pairs.
  - e.g., a pair of semantically similar/dissimilar images may not have feature vectors with relatively small/large Euclidean distance.



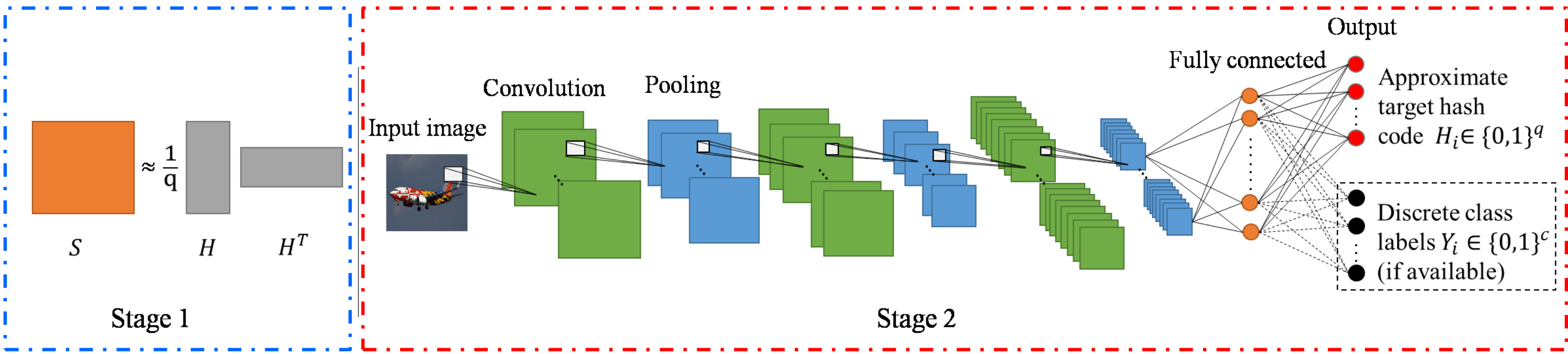
Motivation

A useful image representation is **important** in hash learning process.



The Proposed Approach

Two-stage framework



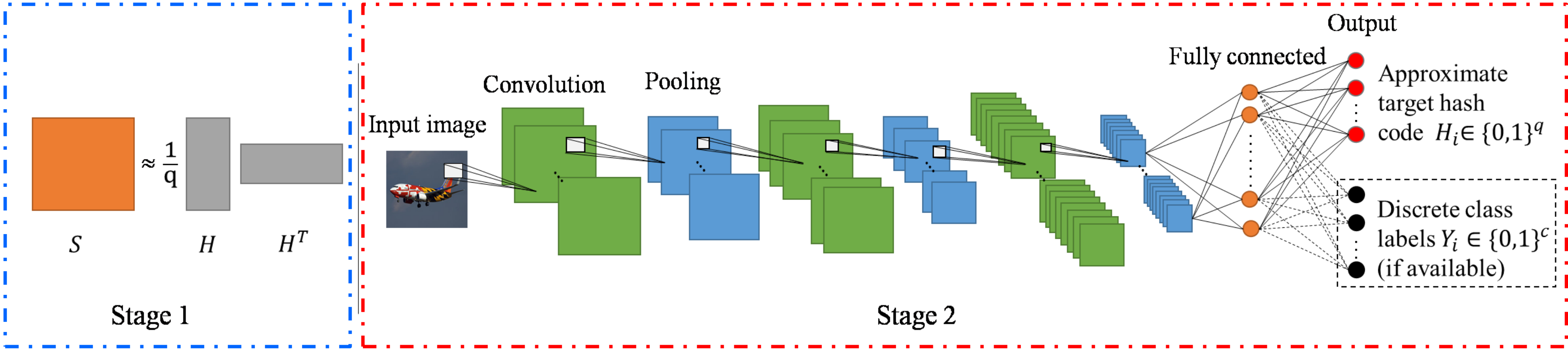
1. **Learn approximate hash codes for the training samples**, i.e., the pairwise similarity matrix S is decomposed into a product $S = \frac{1}{q} H H^T$ where the ith row in **H** is the approximate hash code for the ith training image

$$\min_{H} ||S - \frac{1}{q} H H^T||_F^2 \quad s.t. \; H \in [-1,1]^{n \times q}. \; S_{ij} = \begin{cases} +1, & I_i, I_j \text{ are semantically similar} \\ -1, & I_i, I_j \text{ are semantically dissimilar.} \end{cases}$$

2. **By using the learnt H and the raw image pixels as input, learn image representation and hash functions via** ***deep convolutional neural networks*****.**

The Proposed Approach

Two-stage framework



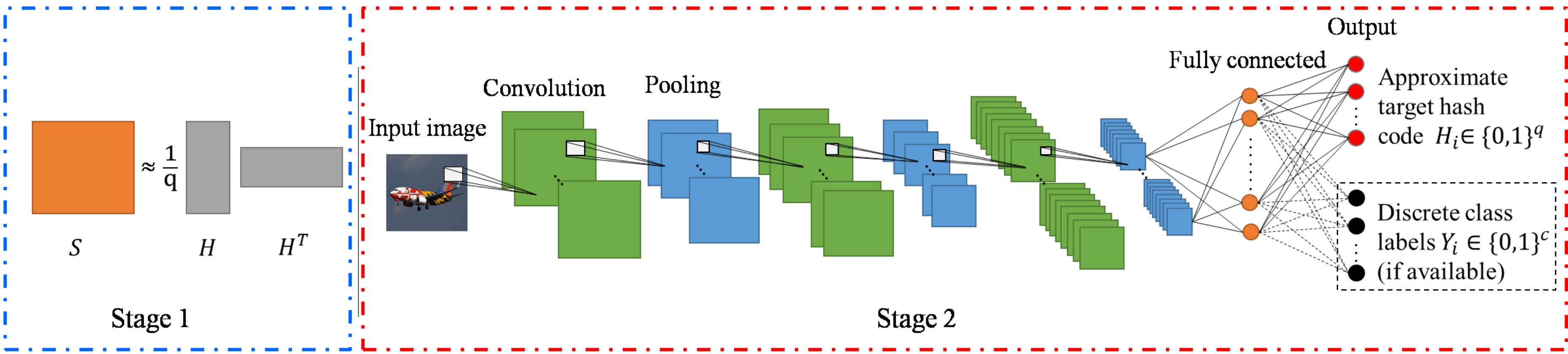
Stage 1

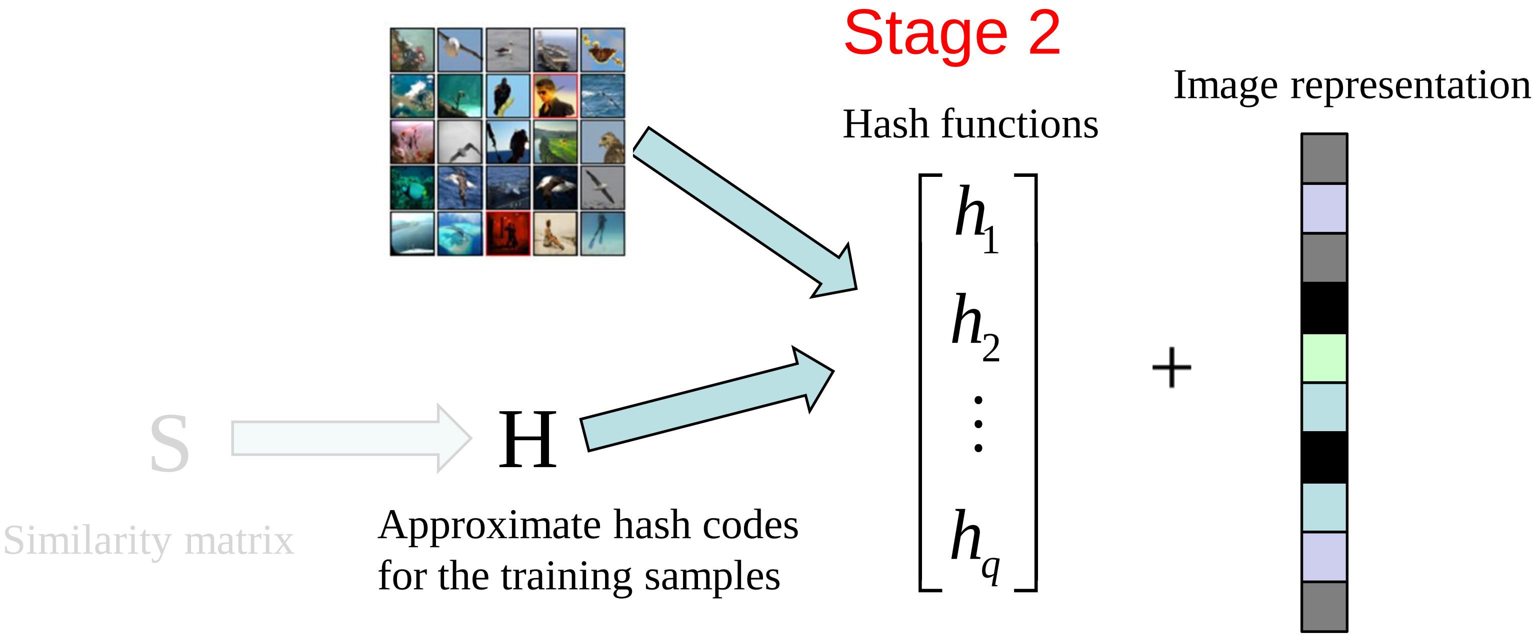
Similarity matrix

Approximate hash codes for the training samples

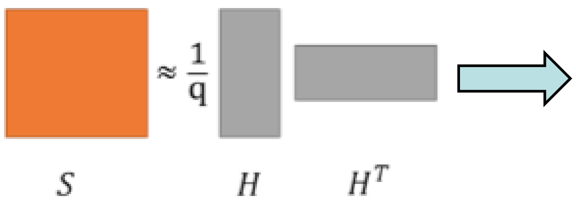
The Proposed Approach

Two-stage framework





Stage 1: Learning Approximate Codes



3 training samples

3-bit Hash codes

$$\underbrace{\begin{pmatrix} 1 & 1 & -1 \\ 1 & 1 & -1 \\ -1 & -1 & 1 \end{pmatrix}}_{\mathbf{S}} \approx \frac{1}{3}\left[\begin{array}{c} x_1 \\ x_2 \\ x_3 \end{array} \underbrace{\begin{bmatrix} 1 & -1 & 1 \\ 1 & -1 & 1 \\ -1 & 1 & -1 \end{bmatrix}}_{\mathbf{H}} \times \underbrace{\begin{array}{ccc} x_1 & x_2 & x_3 \\ \hline 1 & 1 & -1 \\ -1 & -1 & 1 \\ 1 & 1 & -1 \end{array}}_{\mathbf{H}^T} \right]$$

Stage 1

$$\min_H \sum_{i=1}^{n}\sum_{j=1}^{n}(S_{ij} - \frac{1}{q}H_{i\bullet}H_{j\bullet}^T)^2$$

$$= \min_H \| S - \frac{1}{q}HH^T \|_F^2$$

$$subject\ to: H \in \{-1,1\}^{n\times q}$$

The Hamming distance of two hash codes has one-one correspondence to the inner product of these two codes.

If images i and j are **similar** (**dissimilar**), the inner product of their approximate codes should be **large** (**small**).

$$\min_H ||S - \frac{1}{q}HH^T||_F^2 \quad s.t. \boxed{H \in [-1,1]^{n\times q}}$$

relaxation

Optimization

$$\min_{H} ||S - \frac{1}{q} H H^T||_F^2 \;\; s.t.\; H \in [-1, 1]^{n \times q}.$$

- **Algorithm**: random coordinate descent using Newton directions

1. Randomly select an entry H_{ij} in H to update while keeping other entries fixed

2. Approximate the objective function by second-order Taylor expansion w.r.t. H_{ij}
   Calculate a step-size d and update H_{ij} by $H_{ij} \leftarrow H_{ij} + d$

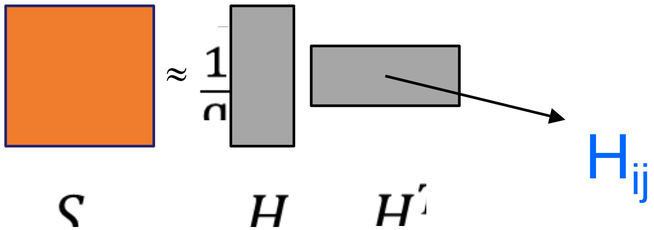
3. Repeat 1 and 2 until stopping criterion is satisfied

Optimization

$$\min_H ||S - \frac{1}{q}HH^T||_F^2 \;\; s.t. \; H \in [-1,1]^{n\times q}.$$

- **Algorithm**: random coordinate descent using Newton directions

1. Randomly select an entry H_{ij} in H to update while keeping other entries fixed



2. Approximate the objective function by second-order Taylor expansion w.r.t. H_{ij}
   Calculate a step-size d and update H_{ij} by $H_{ij} \leftarrow H_{ij} + d$

3. Repeat 1 and 2 until stopping criterion is satisfied

Optimization

$$\min_H \|S - \frac{1}{q} HH^T\|_F^2 \;\; s.t.\; H \in [-1,1]^{n\times q}.$$

- **Algorithm**: random coordinate descent using Newton directions

1. Randomly select an entry H_{ij} in H to update while keeping other entries fixed

2. Approximate the objective function by second-order Taylor expansion w.r.t. H_{ij}
   Calculate a step-size d and update H_{ij} by $H_{ij} \leftarrow H_{ij} + \mathrm{d}$

$$\min_{H_{ij}} g(H_{ij}) = \sum_{l=1}^{n}\sum_{k=1}^{n}(H_{lj}H_{kj} - R_{lk})^2$$
$$= (H_{ij}^2 - R_{ii})^2 + 2\sum_{k\neq i}(H_{ij}H_{kj} - R_{ik})^2 + constant$$
$$subject\ to: H_{ij} \in [-1,1]$$

$$\Longrightarrow \quad \min_d g(H_{ij} + \mathrm{d}) \quad subject\ to: -1 \le H_{ij} + d \le 1$$

$$\Longrightarrow \quad g(H_{ij} + \mathrm{d}) \approx g(H_{ij}) + \mathrm{g}'(H_{ij})d + \frac{1}{2}g''(H_{ij})d^2 \quad \Longrightarrow \quad d = \max(-1 - H_{ij}, \min(-\frac{g'(H_{ij})}{g''(H_{ij})}, 1 - H_{ij}))$$

3. Repeat 1 and 2 until stopping criterion is satisfied

Optimization

$$\min_{H} ||S - \frac{1}{q} H H^T||_F^2 \;\; s.t.\; H \in [-1, 1]^{n \times q}.$$

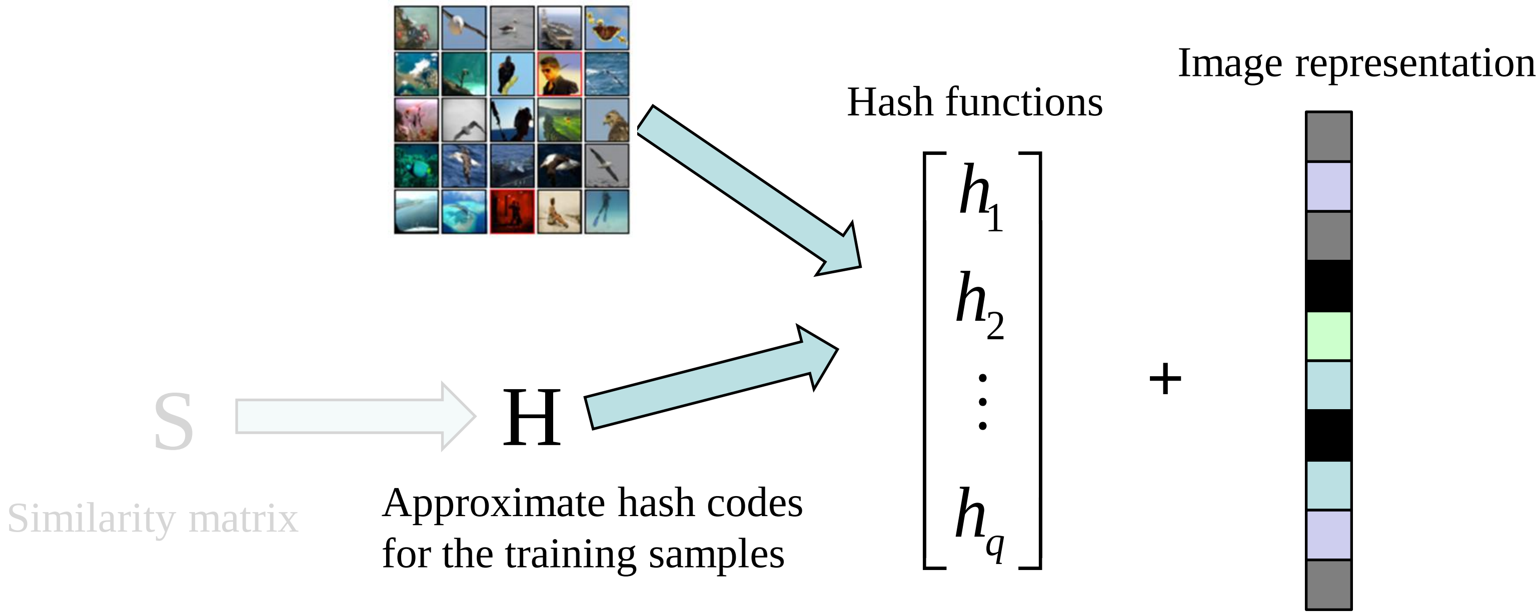
- **Algorithm**: random coordinate descent using Newton directions

1. Randomly select an entry H_{ij} in H to update while keeping other entries fixed
2. Approximate the objective function by second-order Taylor expansion w.r.t. H_{ij}
   Calculate a step-size d and update H_{ij} by $H_{ij} \leftarrow H_{ij} + d$
3. Repeat 1 and 2 until stopping criterion is satisfied

The time complexity of the whole algorithm is O(Tqn^2) with small T, q.

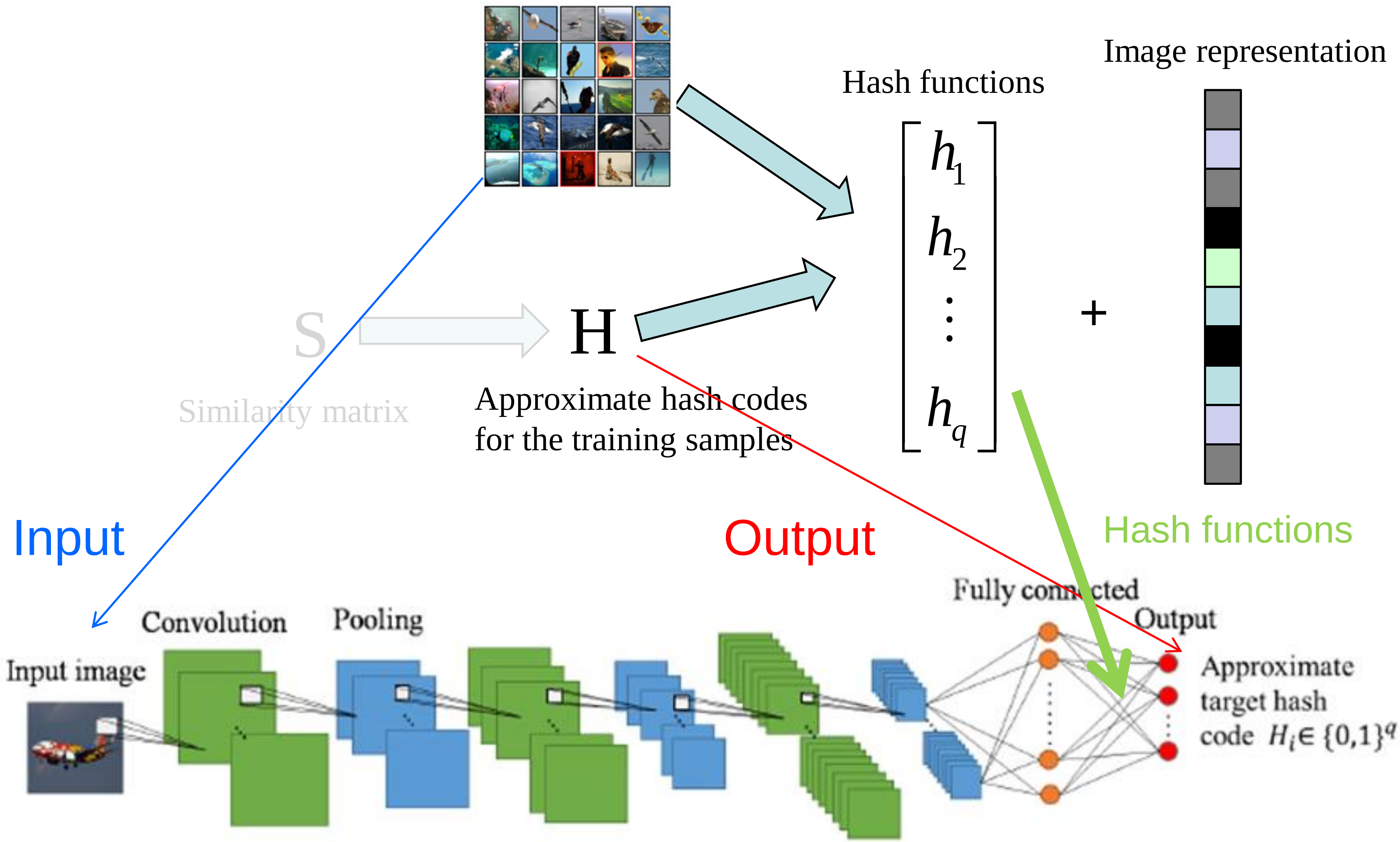
n: the number of training samples
q: hash code length (less than 64 in our experiments)
T: iterations (less than 5 in our experiments)

Stage 2: Learning Hash Functions

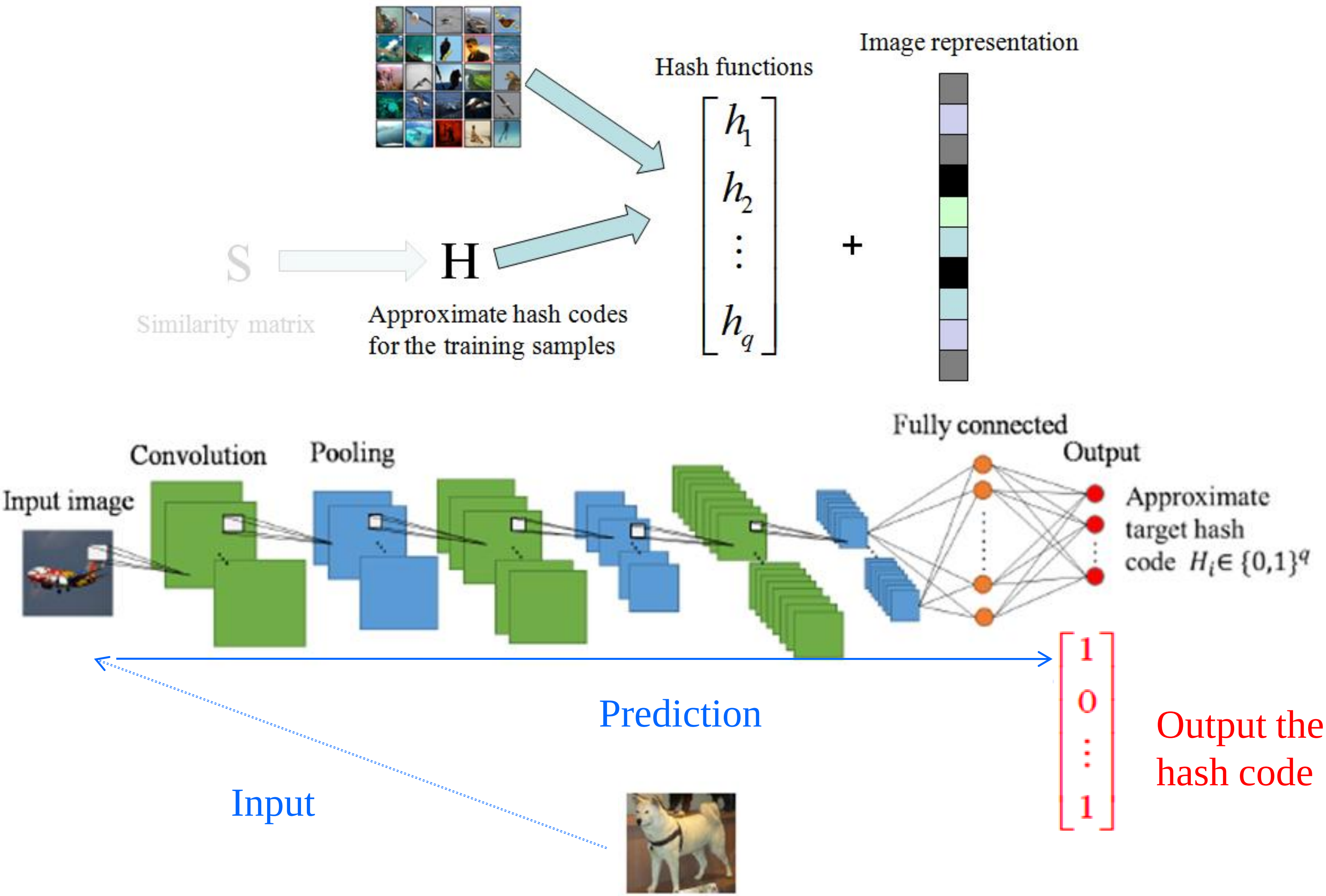


- It is a multi-label binary classification problem that is solved by deep convolutional neural networks.
- It leans hash functions as well as image features.
- We propose two methods: CNNH and CNNH+.

Method 1: CNNH



Method 1: CNNH



Method2: CNNH+

beach, sky, …

(1,0,0,…,1)

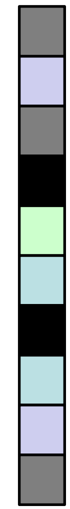
Discrete class labels

Hash functions

$$\begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_q \end{bmatrix}$$

$+$

Image representation

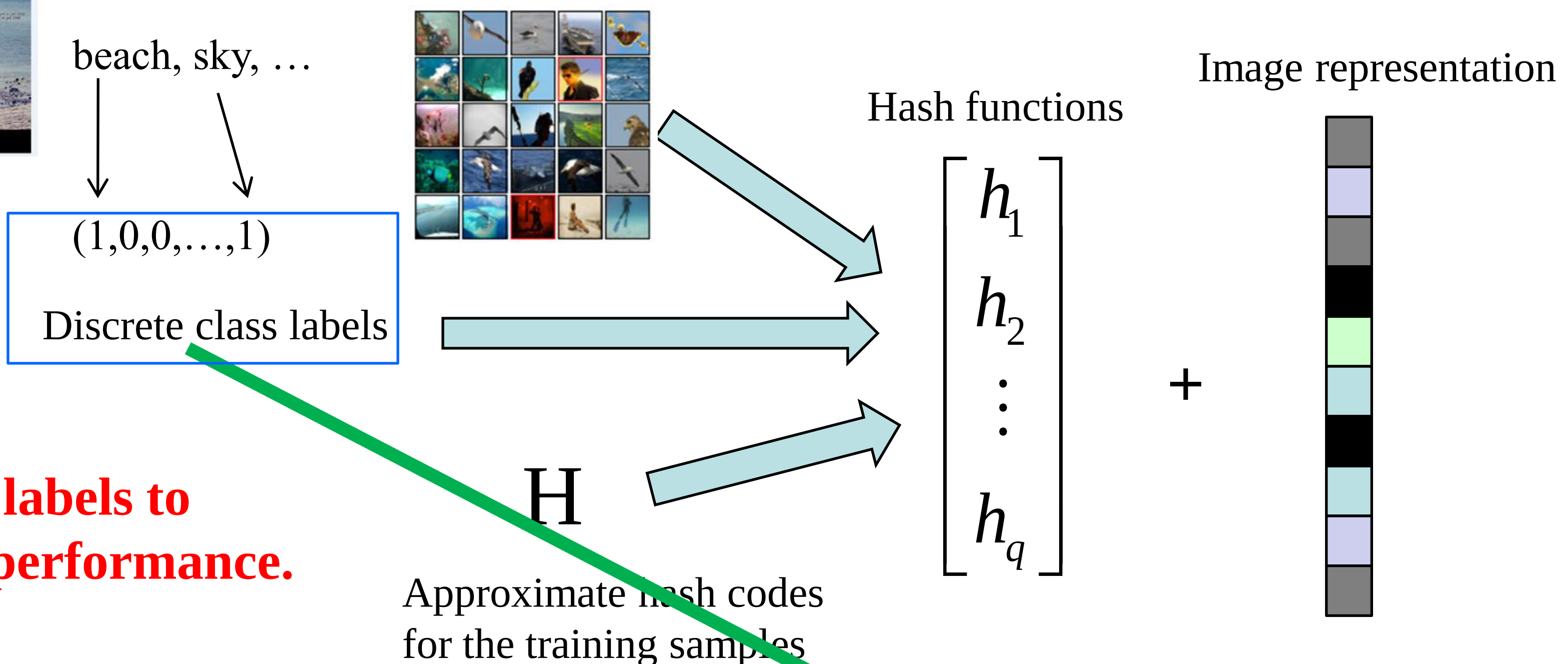


H

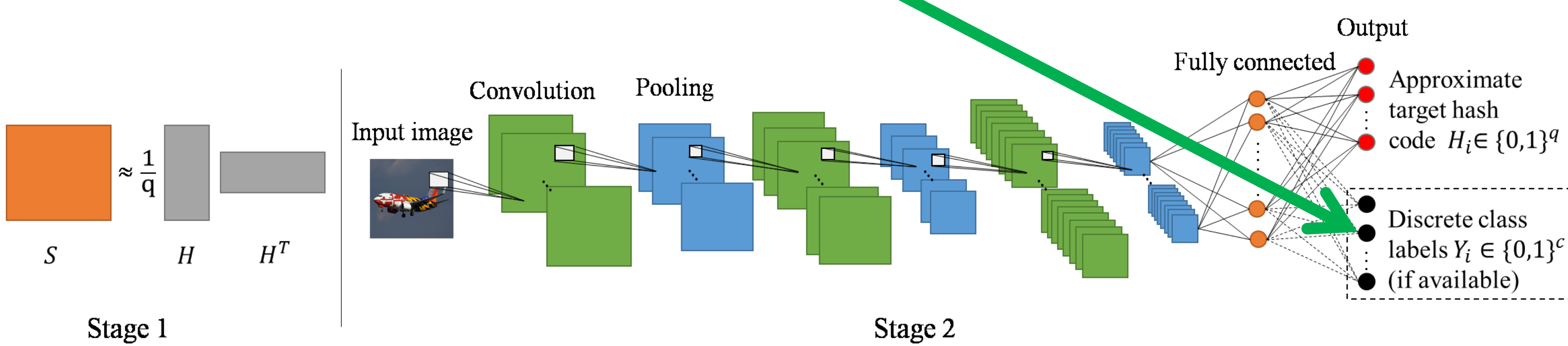
Approximate hash codes
for the training samples

Use class labels to enhance performance.

Method2: CNNH+

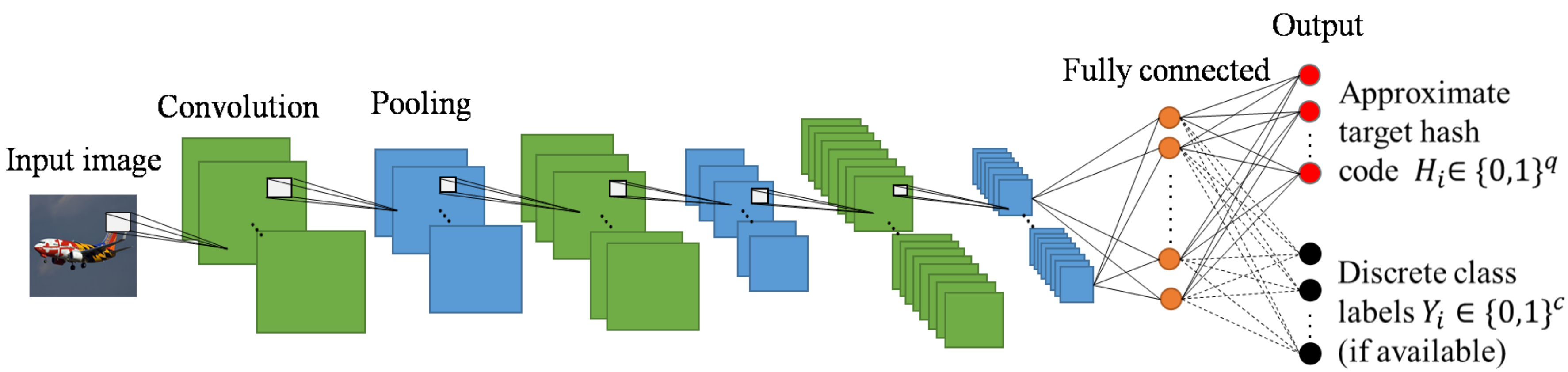


Use class labels to enhance performance.

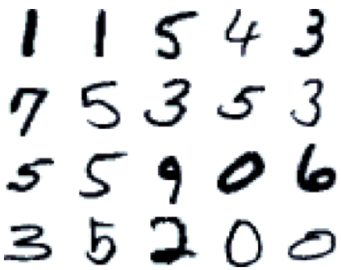


Details of the Deep Convolutional Networks

- We adopt the architecture of [Krizhevsky, NIPS 2012] as our basic framework.
- Our network has three convolutional-pooling layers with rectified linear activation, max pooling and local contrast normalization, a standard fully connected layer, and an output layer with softmax activation.
- We use 32, 64, 128 filter (with the size 5*5) in the 1st, 2nd and 3rd convolutional layer, respectively.
- We use dropout with a rate of 0.5.



Datasets



MNIST: 70,000 greyscale images (in size 28*28) of handwritten digits from ‘0’ to ‘9’

CIFAR10: 60,000 color tinny images (in size 32*32) that are categorized in 10 classes

NUS-WIDE: about 270,000 images collected from the web. It is a multi-label dataset.

Baseline Methods

Unsupervised methods

LSH [Gionis et al. VLDB,1999]

SH [Weiss and Torralba. NIPS,2008]

ITQ [Gong and Lazebnik. CVPR,2011]

MLH [Norouzi and Blei. ICML,2011]

BRE [Kulis and Darrell. NIPS,2009]

ITQ-CCA [Gong and Lazebnik. CVPR,2011]

KSH [Liu et al. CVPR,2012]

Supervised methods

Evaluation Metrics

Precision:

$$precision = \frac{\#\{\text{retrieved relevant images}\}}{\#\{\text{retrieved images}\}}$$

Recall:

$$recall = \frac{\#\{\text{retrieved relevant images}\}}{\#\{\text{all relevant images}\}}$$

Mean Average Precision (MAP):

$$MAP = \frac{1}{q}\sum_i AP_i$$

$$AP = \frac{\sum_n P@n \times I\{\text{image } n \text{ is relevant}\}}{\#\{\text{retrieved relevant image}\}}$$

$$P@n = \frac{\#\{\text{relevant images in top } n \text{ result}\}}{n}$$

Experimental Results

MAP of Hamming ranking on MNIST w.r.t. different number of bits

Methods	12-bit	24-bit	32-bit	48-bit
CNNH+	0.969	0.975	0.971	0.975
CNNH	0.957	0.963	0.956	0.960
KSH	0.872	0.891	0.897	0.900
ITQ-CCA	0.659	0.694	0.714	0.726
MLH	0.472	0.666	0.652	0.654
BRE	0.515	0.593	0.613	0.634
SH	0.265	0.267	0.259	0.250
ITQ	0.388	0.436	0.422	0.429
LSH	0.187	0.209	0.235	0.243

relative increase of 8.2%~11.1%

Experimental Results

MAP of Hamming ranking on CIFAR10 w.r.t. different number of bits

Methods	12-bit	24-bit	32-bit	48-bit
CNNH+	0.465	0.521	0.521	0.532
CNNH	0.439	0.511	0.509	0.522
KSH	0.303	0.337	0.346	0.356
ITQ-CCA	0.264	0.282	0.288	0.295
MLH	0.182	0.195	0.207	0.211
BRE	0.159	0.181	0.193	0.196
SH	0.131	0.135	0.133	0.130
ITQ	0.162	0.169	0.172	0.175
LSH	0.121	0.126	0.120	0.120

relative increase of 49.4%~54.6%

Experimental Results

MAP of Hamming ranking on NUSWIDE w.r.t. different number of bits

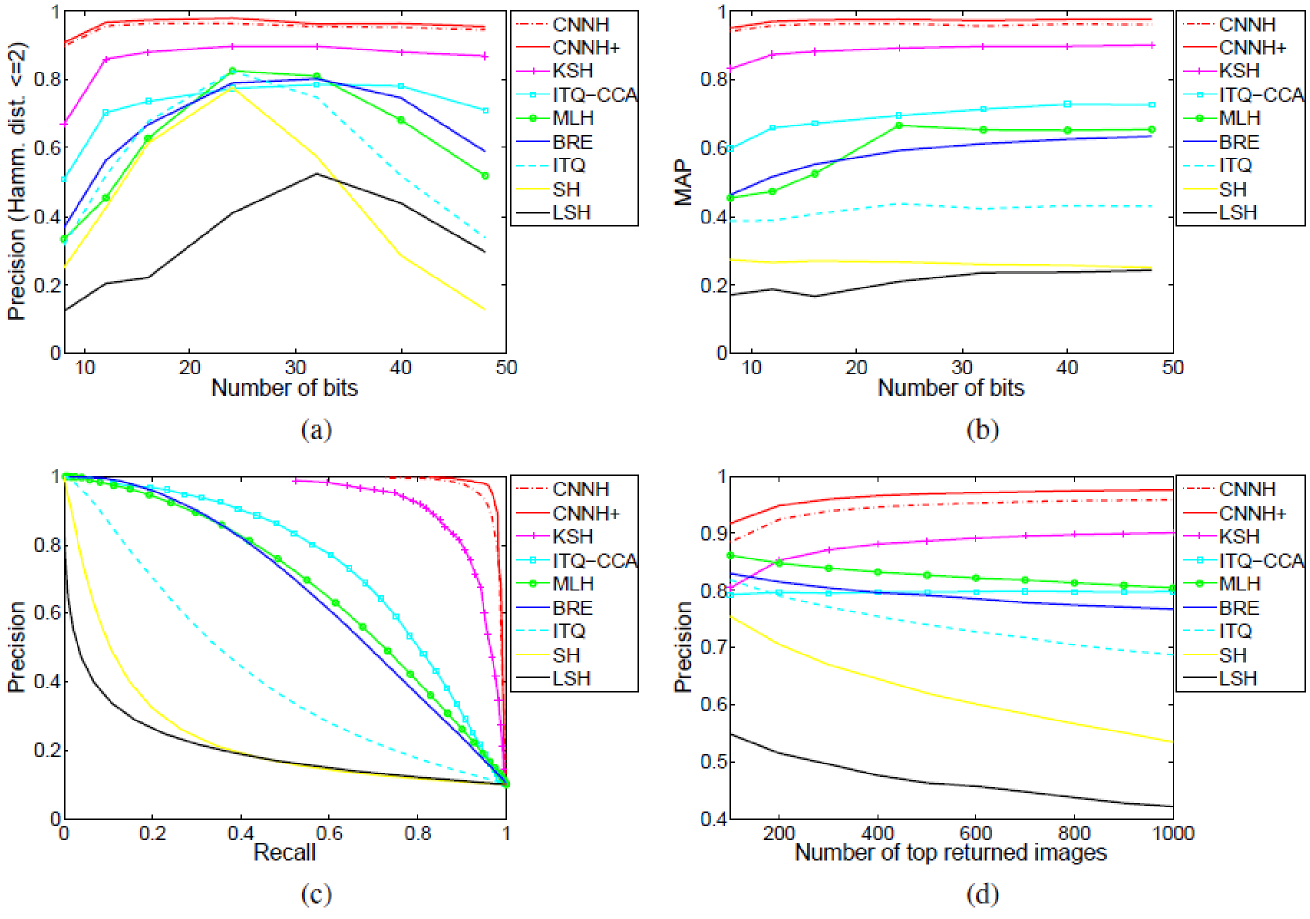
Methods	12-bit	24-bit	32-bit	48-bit
CNNH+	0.623	0.630	0.629	0.625
CNNH	0.611	0.618	0.625	0.608
KSH	0.556	0.572	0.581	0.588
ITQ-CCA	0.435	0.435	0.435	0.435
MLH	0.500	0.514	0.520	0.522
BRE	0.485	0.525	0.530	0.544
SH	0.433	0.426	0.426	0.423
ITQ	0.452	0.468	0.472	0.477
LSH	0.403	0.421	0.426	0.441

relative increase of 6.3%~12.1%

Experimental Results

Results on MNIST

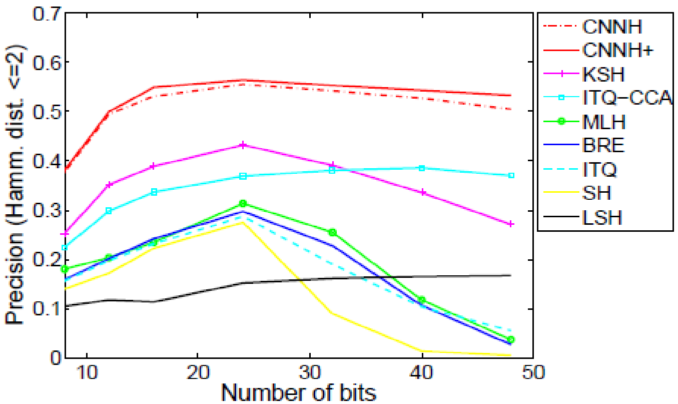
(a) precision within curves Hamming radius 2 (b) MAP curves within Hamming radius 2
(c) precision-recall curves with 48 bits (d) precision curves with 48 bits



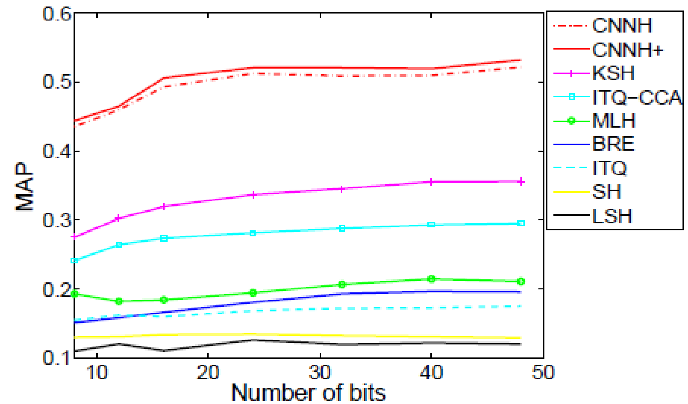
Experimental Results

Results on CIFAR-10

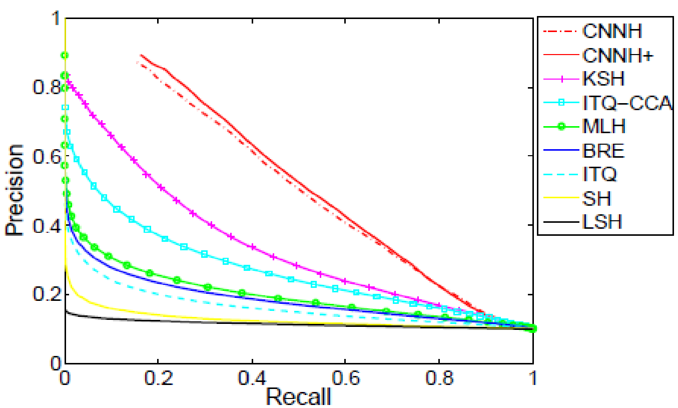
(a) precision curves within Hamming radius 2 (b) MAP curves within Hamming radius 2
(c) precision-recall curves with 48 bits (d) precision curves with 48 bits



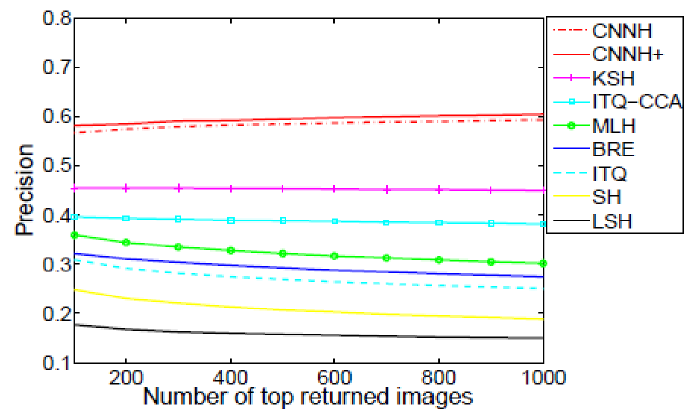
(a)



(b)



(c)

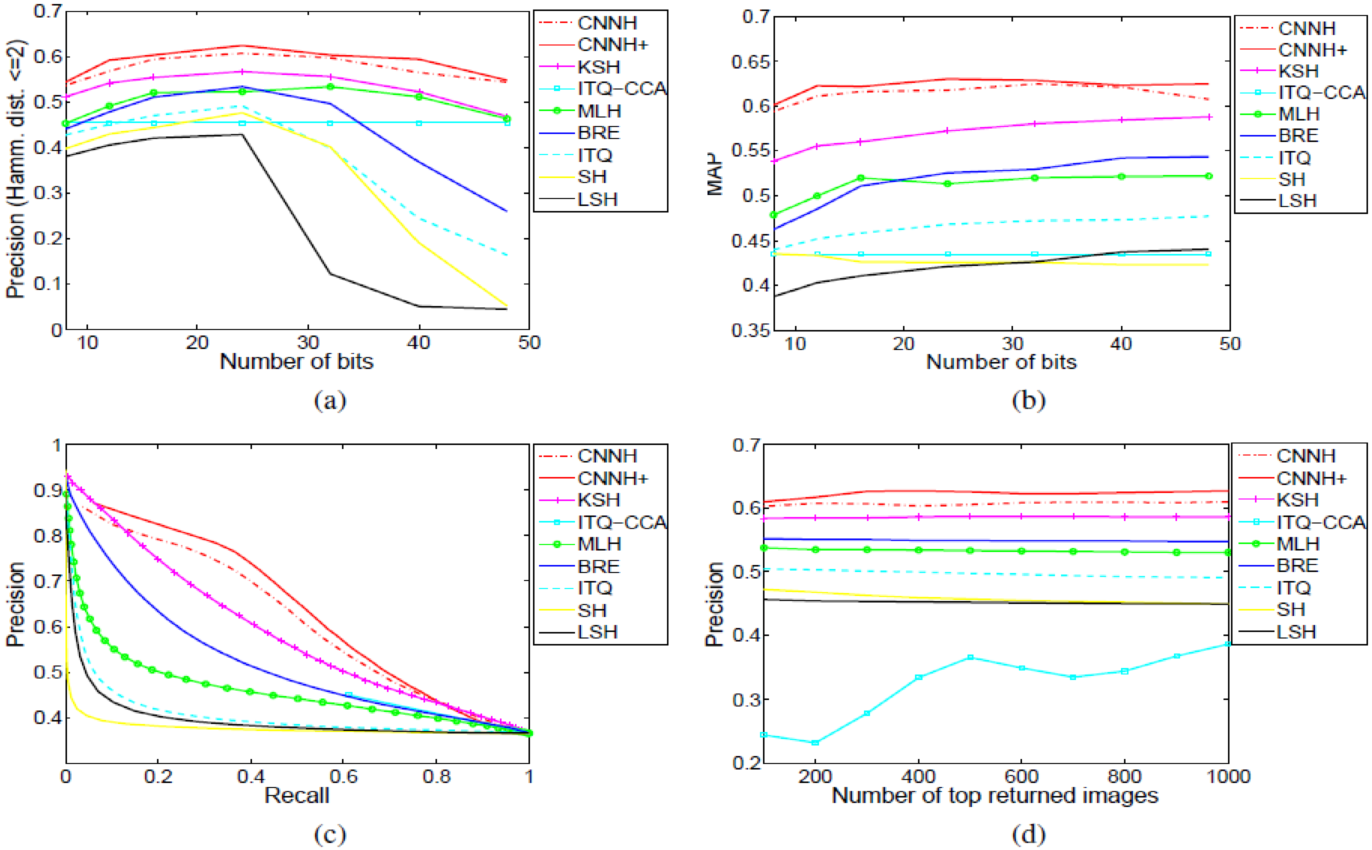


(d)

Experimental Results

Results on NUS-WIDE

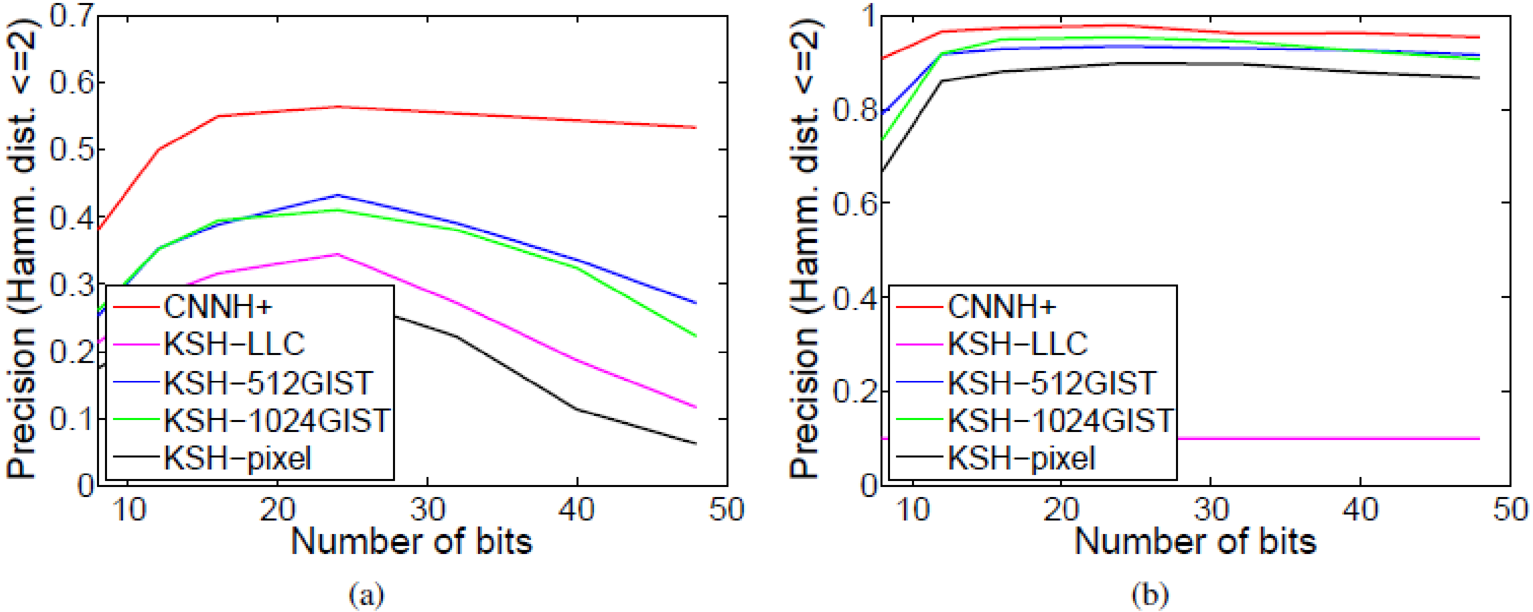
(a) precision curves within Hamming radius 2 (b) MAP curves within Hamming radius 2
(c) precision-recall curves with 48 bits (d) precision curves with 48 bits



Experimental Results

CNNH+ vs. KSH with different hand-crafted features

(a) Results on CIFAR-10 (b) Results on MNIST



The performances of KSH with different features are inferior to those of CNNH+.

Thank you!