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.
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.
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

similar

dissimilar

$x_1$

$x_2$

$x_3$
Similarity-Preserving Hashing

Similarities are well preserved
Similarity-Preserving Hashing

Hashing

Hamming space (2-dim)

Similarities are well preserved

Similarities are not preserved
Related Work

Unsupervised Hashing

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

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]

Long codes are needed to preserve similarities.

Learn compact codes by using label information.
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.

- **Gist vector**
  - 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.

![Diagram showing feature extraction and hashing](image)

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

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

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

Resulting in bad hash codes.
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 $s = \frac{1}{q} HH^T$ where the $i$th row in $H$ is the approximate hash code for the $i$th training image

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

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

S \xrightarrow{\text{Similarity matrix}} H \xrightarrow{\text{Approximate hash codes for the training samples}}
The Proposed Approach

Two-stage framework

Stage 1

Stage 2

Hash functions

Image representation

Approximate hash codes for the training samples
Stage 1: Learning Approximate Codes

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

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

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

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

relaxation

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).
Optimization

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

- **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
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
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 $H_{ij} \leftarrow H_{ij} + d$

\[
\min_{H_{ij}} g(H_{ij}) = \sum_{i=1}^{n} \sum_{k=1}^{n} (H_{ij}H_{kj} - R_{ik})^2
= (H_{ij}^2 - R_{ii})^2 + 2 \sum_{k \neq i} (H_{ij}H_{kj} - R_{ik})^2 + \text{constant}
\]

subject to: $H_{ij} \in [-1, 1]$

\[
\implies g(H_{ij} + d) \approx g(H_{ij}) + g'(H_{ij})d + \frac{1}{2} g''(H_{ij})d^2
\]

\[
\min_{d} g(H_{ij} + d) \quad \text{subject to: } -1 \leq H_{ij} + d \leq 1
\]

3. Repeat 1 and 2 until stopping criterion is satisfied
**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

Input image → Convolution → Pooling → Fully connected → Output

Similarity matrix

Hash functions

Approximate hash codes for the training samples

Image representation

Output

Hash functions

Approximate target hash code $H_t \in \{0,1\}^q$
Method 1: CNNH

Input: Image representation

Output: Hash code

Hash functions: $\begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_q \end{bmatrix}$

Approximate hash codes for the training samples

Fully connected output: $\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$

Prediction: Approximate target hash code $H_t \in \{0,1\}^q$
Method 2: CNNH+

- **Discrete class labels:** beach, sky, …
- **Hash functions:** $h_1, h_2, \ldots, h_q$
- **Image representation**

Use class labels to enhance performance.

Approximate hash codes for the training samples.
Method 2: CNNH+

Discrete class labels

Approximate hash codes for the training samples

Use class labels to enhance performance.
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:

\[ \text{precision} = \frac{\#\{\text{retrieved relevant images}\}}{\#\{\text{retrieved images}\}} \]

Recall:

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

Mean Average Precision (MAP):

\[ \text{MAP} = \frac{1}{q} \sum_{i} AP_i \]

\[ AP = \frac{\sum_{n} P @ n \times I\{\text{image } n \text{ is relevant}\}}{\#\{\text{retrieved relevant image}\}} \]

\[ P @ n = \frac{\#\{\text{relevant images in top } n \text{ result}\}}{n} \]
Experimental Results

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

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

relative increase of 8.2%~11.1%
### Experimental Results

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

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

relative increase of 49.4%~54.6%
### Experimental Results

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

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

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
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!