

# Semi-supervised Hashing with Semantic Confidence for Large Scale Visual Search\*

Yingwei Pan <sup>†</sup>, Ting Yao <sup>‡</sup>, Houqiang Li <sup>†</sup>, Chong-Wah Ngo <sup>#</sup>, Tao Mei <sup>‡</sup>

<sup>†</sup> University of Science and Technology of China, Hefei, China

<sup>‡</sup> Microsoft Research, Beijing, China

<sup>#</sup> City University of Hong Kong, Kowloon, Hong Kong

panyw.ustc@gmail.com, {tiyao, tmei}@microsoft.com; lihq@ustc.edu.cn; cscwngo@cityu.edu.hk

## ABSTRACT

Similarity search is one of the fundamental problems for large scale multimedia applications. Hashing techniques, as one popular strategy, have been intensively investigated owing to the speed and memory efficiency. Recent research has shown that leveraging supervised information can lead to high quality hashing. However, most existing supervised methods learn hashing function by treating each training example equally while ignoring the different semantic degree related to the label, i.e. semantic confidence, of different examples.

In this paper, we propose a novel semi-supervised hashing framework by leveraging semantic confidence. Specifically, a confidence factor is first assigned to each example by neighbor voting and click count in the scenarios with label and click-through data, respectively. Then, the factor is incorporated into the pairwise and triplet relationship learning for hashing. Furthermore, the two learnt relationships are seamlessly encoded into semi-supervised hashing methods with pairwise and listwise supervision respectively, which are formulated as minimizing empirical error on the labeled data while maximizing the variance of hash bits or minimizing quantization loss over both the labeled and unlabeled data. In addition, the kernelized variant of semi-supervised hashing is also presented. We have conducted experiments on both CIFAR-10 (with label) and Clickture (with click data) image benchmarks (up to one million image examples), demonstrating that our approaches outperform the state-of-the-art hashing techniques.

## Categories and Subject Descriptors

H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval—*Retrieval models*

\* This work was performed when Yingwei Pan was visiting Microsoft Research as a research intern.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [Permissions@acm.org](mailto:Permissions@acm.org).

SIGIR'15, August 09 - 13, 2015, Santiago, Chile.

© 2015 ACM. ISBN 978-1-4503-3621-5/15/08 ...\$15.00.

DOI: <http://dx.doi.org/10.1145/2766462.2767725>.

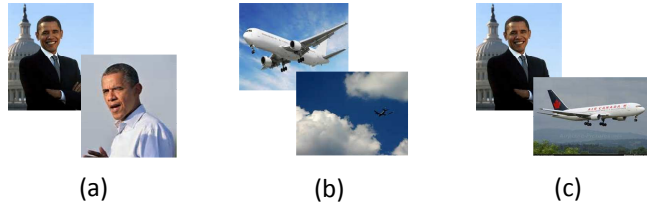


Figure 1: Three exemplary image pairs. (a) Both images are highly relevant to the common label “Barack Obama;” (b) One is highly relevant and the other is weakly relevant to the label “Airplane;” (c) One is highly relevant to the label “Barack Obama,” and the other is highly relevant to the label “Airplane.”

## General Terms

Algorithms, Performance, Experimentation.

## Keywords

Hashing, similarity learning, neighbor voting, semi-supervised hashing, click-through data.

## 1. INTRODUCTION

The rapid development of Web 2.0 technologies has led to the surge of research activities in large scale visual search. One fundamental research problem is similarity search, i.e., nearest neighbor search, which attempts to identify similar instances according to a query example. The need to search for millions of visual examples in a high-dimensional feature space, however, makes the task computationally expensive and thus challenging.

Hashing techniques [23][26][29], one direction of the most well-known Approximate Nearest Neighbor (ANN) search methods, have received intensive research attention for its great efficiency in gigantic data. The basic idea of hashing is to design a compact binary code in a low-dimensional space for each example, making the Hamming distances on similar examples minimized and simultaneously maximized on dissimilar examples. In the similarity search process, the query example is usually first transformed to its hashing code. Then, the Hamming distances between the hash codes of instances in the retrieved database and the query example are computed. The instances within small Hamming distances are returned. Through hashing, the storage is substantially reduced as the representations of examples are highly compressed with a low-dimensional binary space

and the Hamming distance between two hash codes can be implemented by bitwise XOR operation, leading to a very efficient searching process.

While existing hashing approaches are promising to retrieve neighbors, each example is treated equally during the learning of hash function. Importantly, we argue that hashing would obtain higher search accuracy if the semantic degree to the label of each example, i.e., semantic confidence, is taken into account. Figure 1 shows three exemplary image pairs. Conventional supervised hashing methods use to learn hash function to make the corresponding binary codes similar for images annotated by the common label while dissimilar for those with different labels. However, take the two pairs in Figure 1 (a) and (b) as examples, although each pair is relevant to an individual label, the two images in (a) are all highly relevant and thus their hashing code should be closer in proximity than the image pair in (b). Meanwhile, for images that share different labels in Figure 1 (c), their hash codes should be made as dissimilar as possible, especially when they are highly relevant to their respective labels.

By encoding the semantic confidence into hash function learning, this paper presents a novel and principled semi-supervised hashing framework for large scale image search, as illustrated in Figure 2. The semantic confidence is formulated by devising a confidence factor that evaluates the semantic relatedness of an example to a label. We consider two kinds of confidence factors computed based upon image neighbor voting and click count respectively. By assuming the availability of labeled examples, the former first builds a similarity graph over all images of a given label followed by assigning each image a confidence score based on the number of its out-link neighbors. The latter considers click-through data and relates the confidence score directly to the number of click counts an image receives. Based on the semantic confidence, the pairwise and triplet relationships among image examples are further developed and incorporated into the hash function learning in standard (with pairwise supervision) and ranking (with listwise supervision) semi-supervised hashing framework, respectively. In addition, to accommodate linearly inseparable data, a kernel formulation is employed on ranking semi-supervised hashing. After the hash function learning, each image is mapped to compact binary codes. For any query image, the image search list will be returned by sorting their Hamming distances with the query.

In summary, this paper makes the following contributions:

- Instead of assuming each image contributes equally to learning, we explore the learning of a hash function by taking into account the semantic degree to which an image is associated with a label, on the basis that the degree can be quantified as a numeric score (or semantic confidence factor). To the best of knowledge, this paper represents the first effort towards this target.
- By considering two different ways of devising semantic confidences, we propose a general hashing framework that facilitates the exploration of pairwise and triplet image relationships for semi-supervised hashing, ranking-based semi-supervised hashing, and its kernelized variant respectively.
- An extensive set of experiments on two real world image datasets, i.e., CIFAR-10 sampled from 80-million

tiny images collection with labels and Clickture collected from one-year click log from a commercial image search engine, to demonstrate the advantages of the proposed methods over several state-of-the-art hashing techniques.

The remaining sections are organized as follows. Section 2 briefly surveys several popular hashing methods. Section 3 presents the semantic confidence, the pairwise and triplet relationships with semantic confidence, while Section 4 further details the utilization of the two relationships in semi-supervised hashing framework. Section 5 provides empirical evaluations, followed by the discussion and conclusions in Section 6.

## 2. RELATED WORK

We briefly group the related works into three categories: unsupervised, supervised, and semi-supervised hashing.

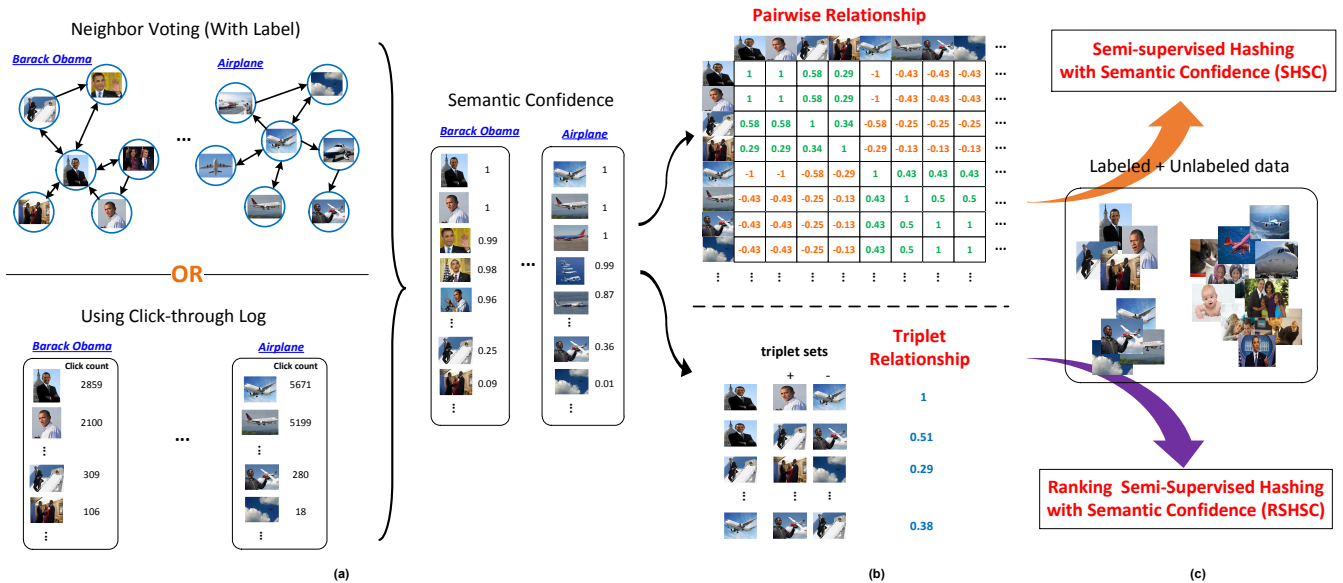
### 2.1 Unsupervised Hashing

Unsupervised hashing refers to the setting when the label information is not available. Locality Sensitive Hashing (LSH) [3] is one of the most popular unsupervised hashing methods, which simply uses random linear projections to construct hash functions. This method was continuously expanded to Kernelized and Multi-Kernel Locality Sensitive Hashing [11][31]. Another effective method called Iterative Quantization (ITQ) [4] was suggested for better quantization rather than random projections. The Spectral Hashing (SH) in [30] was proposed to design compact binary codes by preserving the similarity between samples, which can be viewed as an extension of spectral clustering [33]. Recently, the graph based hashing technique namely Anchor Graph Hashing (AGH) was proposed by Liu *et al.* in [14], which can leverage low-dimensional manifold structure of data to design efficient hash codes. Later in [12], Discrete Graph Hashing (DGH) was proposed to generate high-quality codes by preserving the neighborhood structure of massive data in a discrete code space.

### 2.2 Supervised Hashing

In contrast, when the label information is available, we refer to the problem as supervised hashing. For example, Linear Discriminant Analysis Hashing (LDAH) [24] can tackle supervision via easy optimization. The deep neural network stacked with Restricted Boltzman Machine (RBM) [6] was applied to learn binary hash codes in [23]. To utilize the pairwise supervision information in the hash function learning, Kernel-Based Supervised Hashing (KSH) proposed in [13] used pairwise relationship between samples to achieve high quality hashing. Binary Reconstructive Embedding (BRE) [10] was proposed to learn hash functions by minimizing the reconstructed error between the metric space and Hamming space. Minimal Loss Hashing (MLH) was proposed in [16] which aims to learn similarity-preserving binary codes by using the pairwise relationship. Moreover, there are also several works using the ranking order information to design hash functions. Ranking-based Supervised Hashing (RSH) [27] was proposed to leverage listwise supervision into the hash function learning framework. Besides RSH, the tree-based method [20] and a hamming metric learning framework presented in [17] also aim to preserve the ranking orders.

In addition, several cross-view hashing methods have been proposed as well. The Canonical Correlation Analysis with



**Figure 2: Semi-supervised hashing with semantic confidence.** (a) Semantic confidence measurement of each image example: on the labeled data, the semantic confidence is based on the number of out-link neighbors of each image in the similarity graph built on all images of each label. With the click-through data, click count of an image is considered as an indicator of its semantic confidence in answering the query. (b) Pairwise and Triplet Relationship with semantic confidence: pairwise and triplet relationship is derived by exploiting semantic confidence of each example for hashing with pairwise and listwise supervision, respectively. (c) Semi-supervised hashing: the learnt pairwise and triplet relationship is further incorporated into the hash function learning in standard and ranking semi-supervised hashing framework, respectively. For better viewing, please see original color pdf file.

Iterative Quantization (CCA-ITQ) was proposed in [4] to learn the hash codes from the content features and tags. In [21], Rastegari *et al.* proposed Predictable Dual-View Hashing (PDH) to create a cross-view hamming space by embedding proximity of data samples in the original spaces.

### 2.3 Semi-supervised Hashing

Semi-supervised hashing methods have also been proposed. One representative work is Semi-Supervised Hashing (SSH) [26] which utilizes pairwise information on labeled samples to preserve semantic similarity while remaining robust to overfitting. Semi-Supervised Discriminant Hashing (SSDH) can learn hash codes based on Fisher’s discriminant analysis to maximize the separability between labeled data in different classes while the unlabeled data are used for regularization [8]. In another work [15], the label-regularized maximum margin partition (LAMP) method was proposed to enhance hashing quality by using kernel-based similarity and additional pairwise constraints as side information. Semi-Supervised Tag Hashing (SSTH) proposed in [28] can fully incorporate tag information into hash function learning by exploring the correlation between tags and hashing bits.

In short, our approach belongs to semi-supervised hashing. While these aforementioned semi-supervised hashing methods focus on the regularization of both labeled and unlabeled examples from different means, they treat each image equally and hence the role that each image should play and contribute in learning is overlooked. Our work in this paper contributes by exploring this issue through not only devising the rigorous ways of measuring semantic confidences, but also how the hash function can be more reliably learnt by exploring the semantic confidence.

## 3. LEARNING RELATIONSHIP WITH SEMANTIC CONFIDENCE

In this section, we first define the semantic confidence measurement for each image example, i.e. confidence factor, in the context of scenarios with label and click-through data, respectively. Then, pairwise and triplet relationships with semantic confidence is further devised to be encoded into hash function learning in semi-supervised hashing.

### 3.1 Confidence Factor

Given a label  $t$ , let  $X = \{x_1, x_2, \dots, x_L\}$  be the set of all images annotated by the label  $t$ , where  $x_i$  represents the  $i^{th}$  image in  $X$  and its image feature vector is denoted as  $\mathbf{v}_i$ . Let  $N_k(i)$  be the  $k$ -nearest neighbor set of the  $i^{th}$  image. A directed graph is then built where nodes are all images in  $X$  and there is an edge between the  $i^{th}$  and  $j^{th}$  image if and only if the  $i^{th}$  image appears in  $N_k(j)$ . Deriving from the idea of neighbor voting [34], the semantic confidence of the  $i^{th}$  image is reflected by the number of the image appearing in the neighbors of other images in the graph, i.e., the number of out-link neighbors. Specifically, the confidence factor is generally formulated as

$$s_i = \frac{(o_i^+)^{\gamma}}{\max_j (o_j^+)^{\gamma}}, \quad (1)$$

where  $(o_i^+)$  is the number of out-link neighbors of the  $i^{th}$  image and it is normalized by the maximum number of out-link neighbors of all images in the graph.  $\gamma$  is used to control the impact of the number of out-link neighbors. The rationale underlying this formula is that, if the  $i^{th}$  image always appears in the  $k$ -nearest neighbors of other images, the  $i^{th}$

image is similar to the other images of the category and thus should be more semantically related to this category.

In the scenario with click-through data, we view the click count of an image in response to a query (category) as an indicator of their relevance [19]. As most image search engines display results as thumbnails, the users can view the entire image before clicking on it. As such, the users predominantly tend to click on images that are relevant to the query. Therefore, click count can be served as a reliable connection between queries and images. Then, the confidence factor can be re-expressed as

$$s_i = \frac{(c_i)^\gamma}{\max_j(c_j)^\gamma}, \quad (2)$$

where  $c_i$  is the click count of the  $i^{\text{th}}$  image in response to the query and  $\max_j(c_j)$  is the maximum click count of the returned images for the query. Particularly, the higher the click count, the more the relevance between the image and the query.

### 3.2 Pairwise Relationship with Semantic Confidence

Most of the existing learning to hash methods are to generate hashing code to satisfy pairwise supervision, i.e., making the Hamming distance minimized on similar pairs while maximized on dissimilar pairs. As a result, pairwise relationship is first developed with semantic confidence of each example. Let  $\mathcal{M}$  and  $\mathcal{C}$  be the set of neighbor pairs and non-neighbor pairs, respectively. Specifically, a pair  $(\mathbf{v}_i, \mathbf{v}_j) \in \mathcal{M}$  is denoted as a neighbor pair in which  $\mathbf{v}_i$  and  $\mathbf{v}_j$  are from the same category or in answering the common query. Similarly, a pair  $(\mathbf{v}_i, \mathbf{v}_j) \in \mathcal{C}$  is called as a non-neighbor pair if  $\mathbf{v}_i$  and  $\mathbf{v}_j$  are from the different category or in response to the different query. By incorporating the semantic confidence, the pairwise relationship is defined as

$$S_P(\mathbf{v}_i, \mathbf{v}_j) = \begin{cases} \sqrt{s_i s_j} e^{-|s_i - s_j|}, & \text{if } (\mathbf{v}_i, \mathbf{v}_j) \in \mathcal{M} \\ -\sqrt{s_i s_j} e^{-|s_i + s_j - 2|}, & \text{if } (\mathbf{v}_i, \mathbf{v}_j) \in \mathcal{C} \end{cases}, \quad (3)$$

where  $s_i$  and  $s_j$  are the confidence factor for the  $i^{\text{th}}$  and  $j^{\text{th}}$  image, respectively.

The spirit of pairwise relationship with semantic confidence is to make the neighbor pair in close proximity if the two examples of the pair both have high confidence factors, while weakening the relationship of the neighbor pair if any one in the pair has a low confidence. On the other hand, if the two examples in a non-neighbor pair are both with high confidence to each category, the pair will receive a very dissimilar relationship strengthened by their semantic confidences.

### 3.3 Triplet Relationship with Semantic Confidence

To further leverage the listwise supervision [27] which has been employed to design more effective hash functions in search tasks, semantic confidence is then encoded into triplet relationship learning. Denote  $\mathcal{T}$  as the set of triplets, and each triplet as  $(\mathbf{v}_i, \mathbf{v}_j^+, \mathbf{v}_k^-)$ . In the case when labels are available,  $\mathbf{v}_i$  refers to a query image,  $\mathbf{v}_j^+$  ( $\mathbf{v}_k^-$ ) as the image with the same (different) category as  $\mathbf{v}_i$ . For click data,  $\mathbf{v}_j^+$  refers to the clicked image of the same query on image  $\mathbf{v}_i$ , and  $\mathbf{v}_k^-$  is a clicked image of another query different from

that of  $\mathbf{v}_i$ . For any triplet, we can derive the relationship with semantic confidence as

$$S_T(\mathbf{v}_i, \mathbf{v}_j^+, \mathbf{v}_k^-) = \frac{1}{2} (S_P(\mathbf{v}_i, \mathbf{v}_j^+) - S_P(\mathbf{v}_i, \mathbf{v}_k^-)), \quad (4)$$

where  $S_P(\mathbf{v}_i, \mathbf{v}_j^+)$  and  $S_P(\mathbf{v}_i, \mathbf{v}_k^-)$  are the pairwise relationship defined in Section 3.2. It is straightforward to see that the triplet  $(\mathbf{v}_i, \mathbf{v}_j^+, \mathbf{v}_k^-)$  will be associated with a strong relationship when the image  $\mathbf{v}_j^+$  is close to the query image  $\mathbf{v}_i$ , and simultaneously the image  $\mathbf{v}_k^-$  holds a quite dissimilar relationship with the image  $\mathbf{v}_i$ .

## 4. SEMI-SUPERVISED HASHING

In this section, we will present our semi-supervised hashing framework under the umbrella of encoding the learnt relationships with semantic confidence to three primary aspects: semi-supervised hashing by exploiting pairwise relationship with semantic confidence, ranking semi-supervised hashing by leveraging triplet relationship with semantic confidence and its kernelized variant.

Suppose there are  $n$  images in the whole set, represented as:  $V = \{\mathbf{v}_i | i = 1, \dots, n\}$ , where  $\mathbf{v}_i \in \mathbb{R}^D$  represents the image feature vector and  $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n\} \in \mathbb{R}^{D \times n}$  is the feature matrix of the image set. Similarly, assume there are  $L$  ( $L < n$ ) labeled images and the feature matrix of the labeled images are denoted as  $\mathbf{V}_l \in \mathbb{R}^{D \times L}$ . Note that the feature matrices are normalized to zero-centered. Our goal is to map  $\mathbf{V} \in \mathbb{R}^{D \times n}$  to a compact binary code representation  $\mathbf{B} \in \{-1, 1\}^{K \times n}$  in a low-dimensional Hamming space, where  $K$  is the code length.

### 4.1 Semi-supervised Hashing with Semantic Confidence (SHSC)

Here we use the linear formulation to design the hashing functions. For each bit  $k = 1, \dots, K$ , its hash function is defined as

$$h_k(\mathbf{v}_i) = \text{sgn}(\mathbf{w}_k \cdot \mathbf{v}_i), \quad (5)$$

where  $\mathbf{w}_k \in \mathbb{R}^D$  is the coefficient vector and  $\text{sgn}(\bullet)$  is the signum function. Let  $\mathbf{W} = \{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_K\}^\top \in \mathbb{R}^{K \times D}$  be the projection matrix, we can get the  $K$ -bit binary code representation  $\mathbf{B}$  of an image set  $\mathbf{V}$  as

$$\mathbf{B} = \text{sgn}(\mathbf{W}\mathbf{V}). \quad (6)$$

Inspired by the idea of semi-supervised hashing [26], we propose the semi-supervised hashing with semantic confidence. The problem is formulated as simultaneously maximizing empirical accuracy on the labeled images and the variance of hash bits over both the labeled and unlabeled images, in which pairwise relationship with semantic confidence is encoded into the computation of empirical accuracy. Specifically, the empirical accuracy on the label images is defined as

$$J_1(\mathbf{W}) = \sum_k \left( \sum_{(\mathbf{v}_i, \mathbf{v}_j) \in \mathcal{M}} S_P(\mathbf{v}_i, \mathbf{v}_j) h_k(\mathbf{v}_i) h_k(\mathbf{v}_j) + \sum_{(\mathbf{v}_i, \mathbf{v}_j) \in \mathcal{C}} S_P(\mathbf{v}_i, \mathbf{v}_j) h_k(\mathbf{v}_i) h_k(\mathbf{v}_j) \right), \quad (7)$$

where  $S_P(\mathbf{v}_i, \mathbf{v}_j)$  is the pairwise relationship with semantic confidence for pair  $(\mathbf{v}_i, \mathbf{v}_j)$ . By defining the pairwise relationship matrix  $\mathbf{S} \in \mathbb{R}^{L \times L}$  on the labeled images  $\mathbf{V}_l$  with

its element  $\mathbf{S}_{i,j} = S_p(\mathbf{v}_i, \mathbf{v}_j)$ , the empirical accuracy  $J_1(\mathbf{W})$  can be represented as

$$J_1(\mathbf{W}) = \frac{1}{2} \text{tr} \left( \text{sgn}(\mathbf{W}\mathbf{V}_l) \mathbf{S} \text{sgn}(\mathbf{W}\mathbf{V}_l)^\top \right). \quad (8)$$

Through maximizing empirical accuracy on the labeled images, the hash codes will be in close proximity for neighboring pairs with high pairwise relationship, while very different for non-neighboring pairs especially when both samples are with high confidence to different category.

On the other hand, to generate hash codes in which each bit maximizes the information by generating a balanced partition of the data, the variance of hash bits should be also maximized. Here, the variance of hash bits over the labeled and unlabeled images is measured as

$$J_2(\mathbf{W}) = \frac{1}{2} \text{tr} \left( \mathbf{W}\mathbf{V}(\mathbf{W}\mathbf{V})^\top \right). \quad (9)$$

The overall objective function integrates the empirical accuracy on the labeled images and the variance of hash bits over the labeled and unlabeled images. Hence we get the following optimization problem

$$\max_{\mathbf{W}} J_1(\mathbf{W}) + \mu J_2(\mathbf{W}) \quad \text{s.t.} \quad \mathbf{W}\mathbf{W}^\top = \mathbf{I}, \quad (10)$$

where  $\mu$  is the tradeoff parameter and the constraint  $\mathbf{W}\mathbf{W}^\top = \mathbf{I}$  limits the hashing projection matrix to be orthogonal.

We use non-orthogonal projection learning [26] for the optimization, which relaxes the orthogonality constraint and solves the non-convex problem with matrix decomposition.

## 4.2 Ranking SHSC

To design more effective hash functions, we further incorporate the triplet relationship with semantic confidence into ranking semi-supervised hashing which represents the ranking information by a set of ranking triplets. The training of Ranking SHSC (RSHSC) is performed by minimizing both the triplet loss based on the labeled images and the quantization loss on the whole image set.

**Formulation.** For Ranking SHSC, we still use the linear form of hash functions as defined in Eq.(5). Formally, given an image pair  $(\mathbf{v}_i, \mathbf{v}_j)$ , we revise a distance function to measure the degree of hash code difference as

$$d(\mathbf{v}_i, \mathbf{v}_j) = (\text{sgn}(\mathbf{W}\mathbf{v}_i) - \text{sgn}(\mathbf{W}\mathbf{v}_j))^\top (\text{sgn}(\mathbf{W}\mathbf{v}_i) - \text{sgn}(\mathbf{W}\mathbf{v}_j)). \quad (11)$$

The triplet loss on the labeled images  $\mathbf{V}_l$  is defined as

$$J_1(\mathbf{W}) = \sum_{(\mathbf{v}_i, \mathbf{v}_j^+, \mathbf{v}_k^-) \in \mathcal{T}} S_T(\mathbf{v}_i, \mathbf{v}_j^+, \mathbf{v}_k^-) L(\mathbf{v}_i, \mathbf{v}_j^+, \mathbf{v}_k^-), \quad (12)$$

$$L(\mathbf{v}_i, \mathbf{v}_j^+, \mathbf{v}_k^-) = \max(0, d(\mathbf{v}_i, \mathbf{v}_j^+) - d(\mathbf{v}_i, \mathbf{v}_k^-) + 1),$$

where  $S_T(\mathbf{v}_i, \mathbf{v}_j^+, \mathbf{v}_k^-)$  is the triplet relationship with semantic confidence consisting of the query image  $\mathbf{v}_i$ , an image  $\mathbf{v}_j^+$  from the same category and an image  $\mathbf{v}_k^-$  from a different category. Note that all the triplets in Eq.(12) are from the triplet sets  $\mathcal{T}$  generated on labeled images  $\mathbf{V}_l$ . The triplet loss exploits the margin ranking loss [5][32] that is widely used in information retrieval weighted by the triplet relationship. By minimizing the triplet loss on the labeled images, the relative distance relationship for hash codes in hamming space is preserved under the listwise supervision. Specifically, for the triplet  $(\mathbf{v}_i, \mathbf{v}_j^+, \mathbf{v}_k^-)$  with strong relationship, we aim to make the hash codes in close proximity

for  $\mathbf{v}_i$  and  $\mathbf{v}_j^+$ , and simultaneously get the very different bits for  $\mathbf{v}_i$  and  $\mathbf{v}_k^-$ .

To better generate hash codes and avoid overfitting, we additionally incorporate another term by using all the labeled and unlabeled images, leading to a semi-supervised framework. Motivated by iterative quantization [4], the quantization loss is defined on the whole image set  $\mathbf{V}$  as

$$J_2(\mathbf{W}) = \|\mathbf{B} - \mathbf{W}\mathbf{V}\|_F^2, \quad (13)$$

where  $\mathbf{B} = \text{sgn}(\mathbf{W}\mathbf{V})$  and  $\|\bullet\|_F$  denotes the Frobenius norm. Minimization of the quantization loss will preserve the original locality structure better in the generated hash codes.

The overall objective function for Ranking SHSC is comprised of the triplet loss in Eq.(12) and the quantization loss in Eq.(13). Hence we get the following optimization problem for our RSHSC:

$$\min_{\mathbf{W}} \sum_{(\mathbf{v}_i, \mathbf{v}_j^+, \mathbf{v}_k^-) \in \mathcal{T}} S_T(\mathbf{v}_i, \mathbf{v}_j^+, \mathbf{v}_k^-) L(\mathbf{v}_i, \mathbf{v}_j^+, \mathbf{v}_k^-) + \alpha \|\mathbf{B} - \mathbf{W}\mathbf{V}\|_F^2 \quad \text{s.t.} \quad \mathbf{W}\mathbf{W}^\top = \mathbf{I}, \quad (14)$$

where  $\alpha$  is the tradeoff parameter and the constraint  $\mathbf{W}\mathbf{W}^\top = \mathbf{I}$  forces the hashing projection matrix  $\mathbf{W}$  to be orthogonal, making the bits of the generated hash codes uncorrelated to each other.

**Optimization.** The orthogonal constraints and the non-differentiable terms (i.e.,  $\text{sgn}(\bullet)$ ) in Eq.(14) make the optimization difficult to be solved. To address this problem, we first relax the overall objective function by replacing the signum function in Eq.(11) with its signed magnitude as suggested in [26][28]. With this relaxation, the distance function in Eq.(11) can be rewritten as

$$d(\mathbf{v}_i, \mathbf{v}_j) = (\mathbf{W}(\mathbf{v}_i - \mathbf{v}_j))^\top \mathbf{W}(\mathbf{v}_i - \mathbf{v}_j). \quad (15)$$

Thus, the triplet loss in Eq.(12) becomes differentiable.

Next, the orthogonal constraint  $\mathbf{W}\mathbf{W}^\top = \mathbf{I}$  can be relaxed by appending the converted soft penalty term to the objective function. The penalty term is defined as

$$J_3(\mathbf{W}) = \|\mathbf{W}\mathbf{W}^\top - \mathbf{I}\|_F^2. \quad (16)$$

After the two relaxations, the overall objective function becomes

$$\min_{\mathbf{W}} \sum_{(\mathbf{v}_i, \mathbf{v}_j^+, \mathbf{v}_k^-) \in \mathcal{T}} S_T(\mathbf{v}_i, \mathbf{v}_j^+, \mathbf{v}_k^-) L(\mathbf{v}_i, \mathbf{v}_j^+, \mathbf{v}_k^-) + \alpha \|\mathbf{B} - \mathbf{W}\mathbf{V}\|_F^2 + \beta \|\mathbf{W}\mathbf{W}^\top - \mathbf{I}\|_F^2, \quad (17)$$

where  $\alpha$  and  $\beta$  are the tradeoff parameters.

To address the relaxed optimization problem in Eq.(17), the stochastic gradient descent is used for its efficiency and capability in dealing with highly scalable problems. Note that the quantization loss term in Eq.(13) cannot be relaxed by its signed magnitude directly, otherwise this quantization loss will go to 0, which is meaningless in practice. Similar to the common solution used in [4][28], in each iteration of the gradient descent procedure, we split the optimization process into two steps: 1) fix  $\mathbf{W}$  and update  $\mathbf{B} = \text{sgn}(\mathbf{W}\mathbf{V})$ ; 2) fix  $\mathbf{B}$  and update  $\mathbf{W}$  according to the gradient descent for the objective function. We alternate the process of updating  $\mathbf{B}$  and  $\mathbf{W}$  to find a locally optimal solution. The whole RSHSC algorithm is given in Algorithm 1.

---

**Algorithm 1** Ranking Semi-supervised Hashing with Semantic Confidence (RSHSC)

---

- 1: **Input:** Training images  $\mathcal{V}$  and labeled images  $\mathbf{V}_l$ .
  - 2: Generate a set of triplets  $\mathcal{T}$  consisting of  $(\mathbf{v}_i, \mathbf{v}_j^+, \mathbf{v}_k^-)$  from the labeled images  $\mathbf{V}_l$ .  
Initialize the projection matrix  $\mathbf{W}$  using a normal distribution with mean zero and standard deviation one.  
Initialize the learning rate  $\eta$ , and two tradeoff parameters  $\alpha$  and  $\beta$ .
  - 3: **for**  $iter = 1$  to  $T_{\max}$  **do**
  - 4:   Select a random triplet  $(\mathbf{v}_i, \mathbf{v}_j^+, \mathbf{v}_k^-)$  from  $\mathcal{T}$ .
  - 5:    $\mathbf{B} = \text{sgn}(\mathbf{W}\mathbf{V})$
  - 6:    $\mathbf{W} = \mathbf{W} - \eta(-2\alpha(\mathbf{B} - \mathbf{W}\mathbf{V})\mathbf{V}^\top + 2\beta(\mathbf{W}\mathbf{W}^\top - \mathbf{I})\mathbf{W})$
  - 7:    $L(\mathbf{v}_i, \mathbf{v}_j^+, \mathbf{v}_k^-) = \max(0, d(\mathbf{v}_i, \mathbf{v}_j^+) - d(\mathbf{v}_i, \mathbf{v}_k^-) + 1)$
  - 8:   **if**  $L(\mathbf{v}_i, \mathbf{v}_j^+, \mathbf{v}_k^-) \succ 0$  **then**
  - 9:      $\nabla L = 2\mathbf{W}(\mathbf{v}_i - \mathbf{v}_j^+)(\mathbf{v}_i - \mathbf{v}_j^+)^\top - 2\mathbf{W}(\mathbf{v}_i - \mathbf{v}_k^-)(\mathbf{v}_i - \mathbf{v}_k^-)^\top$
  - 10:     Compute  $S_T(\mathbf{v}_i, \mathbf{v}_j^+, \mathbf{v}_k^-)$  via Eq.(4).
  - 11:      $\mathbf{W} = \mathbf{W} - \eta S_T(\mathbf{v}_i, \mathbf{v}_j^+, \mathbf{v}_k^-)\nabla L$
  - 12:   **end if**
  - 13: **end for**
  - 14: **Output:**  
The optimized hashing projection matrix  $\mathbf{W}$ .
- 

### 4.3 Kernel-based Ranking SHSC

Our Ranking SHSC method can be easily kernelized (Kernel-based Ranking SHSC) through the kernel trick which has been proven to be able to tackle linearly inseparable data. To kernelize Ranking SHSC, we use a kernel function  $\kappa : \mathbb{R}^D \times \mathbb{R}^D \mapsto \mathbb{R}$  to construct the hash functions. Following the kernel-based hashing algorithms [11][13], we define the kernelized hash function with the kernel  $\kappa$  plugged in as

$$\tilde{h}(\mathbf{v}_i) = \text{sgn} \left( \sum_{j=1}^m \kappa(\mathbf{v}_{(j)}, \mathbf{v}_i) w_j - b \right), \quad (18)$$

where  $\mathbf{v}_{(1)}, \dots, \mathbf{v}_{(m)}$  are  $m$  samples randomly selected from  $\mathbf{V}$ ,  $w_j \in \mathbb{R}$  is the coefficient and  $b \in \mathbb{R}$  is the bias. It is worth noting that in order to make this kernel-based hashing fast,  $m$  is set to be much smaller than the image dataset size  $n$ . Following the balancing criterion [4][13] in hash function that the generated hash bit should take as much information as possible, the bias  $b$  is set as

$$b = \sum_{i=1}^n \sum_{j=1}^m \kappa(\mathbf{v}_{(j)}, \mathbf{v}_i) w_j / n. \quad (19)$$

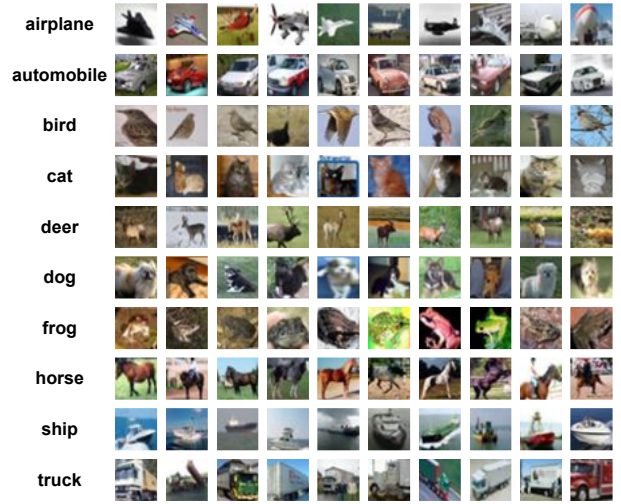
Therefore, the kernelized hash function is rewritten as

$$\begin{aligned} \tilde{h}(\mathbf{v}_i) &= \text{sgn} \left( \sum_{j=1}^m (\kappa(\mathbf{v}_{(j)}, \mathbf{v}_i) - \frac{1}{n} \sum_{i=1}^n \kappa(\mathbf{v}_{(j)}, \mathbf{v}_i)) w_j \right) \\ &= \text{sgn}(\mathbf{w}\bar{\kappa}(\mathbf{v}_i)), \end{aligned} \quad (20)$$

where  $\mathbf{w} = [w_1, \dots, w_m]$  is the coefficient vector and  $\bar{\kappa} : \mathbb{R}^D \mapsto \mathbb{R}^m$  is the vectorial map, which is defined as

$$\begin{aligned} \bar{\kappa}(\mathbf{v}_i) &= [\kappa(\mathbf{v}_{(1)}, \mathbf{v}_i) - b_1, \dots, \kappa(\mathbf{v}_{(m)}, \mathbf{v}_i) - b_m]^\top, \\ b_j &= \frac{1}{n} \sum_{i=1}^n \kappa(\mathbf{v}_{(j)}, \mathbf{v}_i). \end{aligned} \quad (21)$$

With the vectorial map  $\bar{\kappa}$ , we can obtain the kernelized feature matrix  $\mathbf{V}_\kappa \in \mathbb{R}^{m \times n}$ . The  $K$ -bit kernel-based binary



**Figure 3:** Ten example images randomly selected from each class in CIFAR-10 dataset. The class label is given in the first column.

code representation  $\mathbf{B}_\kappa$  of the image set  $\mathbf{V}$  is then given as

$$\mathbf{B}_\kappa = \text{sgn}(\mathbf{W}\mathbf{V}_\kappa), \quad (22)$$

where  $\mathbf{W}$  is learnt by using the proposed Ranking SHSC method.

## 5. EXPERIMENTS

We conducted large-scale image retrieval experiments on two image datasets, i.e., CIFAR-10<sup>1</sup>, a tiny image collection in 10 classes and Clickture [7], a click-based image dataset.

### 5.1 Datasets

The **CIFAR-10** dataset contains 60,000 real world tiny images ( $32 \times 32$  pixels), which is a labeled subset of the 80 million tiny images dataset [25]. It consists of 10 object classes and each class contains 6K samples. Every image in this dataset is assigned to a mutually exclusive class label and represented by a 512-dimensional GIST feature vector [18]. Figure 3 shows 10 randomly selected images from each class in CIFAR-10.

The dataset is partitioned into two parts: a training set with 59K images and a test set with 1K images evenly sampled from ten classes. We additionally sample 100 images from each class in the training set and constitute 1K labeled subset for training. For each test image, the ground-truth similar images are derived from class label, i.e., images from the same class are deemed to be similar.

**Clickture** is a large-scale click based image dataset [7]. It was collected from one year click-through data of one commercial image search engine. The dataset comprises two parts, i.e., the training and development (dev) sets. The training set consists of 23.1 million  $\{query, image, click\}$  triads of 11.7 millions distinct queries and 1.0 million unique images. Figure 4 shows a few exemplary queries with their clicked images and click counts in the Clickture. For example, users clicked the first image 25 times in the search results

<sup>1</sup><http://www.cs.toronto.edu/~kriz/cifar.html>








cardinal logo	red fox	sun moon	leaf
 25	 983	 20	 673
 13	 306	 13	 518
 6	 20	 9	 48
 2	 12	 5	 1
 1	 1	 2	 1

Figure 4: Examples in Clickture dataset (upper row: search queries; lower row: images with click times in response to the upper query).

when submitting query “cardinal logo” in total. In the dev dataset, there are 79,926  $\langle query, image \rangle$  pairs generated from 1,000 queries, where each image to the corresponding query was manually annotated on a three point ordinal scale: Excellent, Good, and Bad. Inspired by the success of deep convolutional neural networks (DCNN) [1][9], we take the output of 1000-way fc8 classification layer by using DeCAF [2] as the image representation for Clickture dataset, which constitutes a 1000-dimensional feature vector.

In the experiment, we adopt 1.0 million unique images in the training set as our training data and randomly sampled 10K images as labeled subset. Moreover, 1K unique images that are annotated as “Excellent” to the query in the dev set are randomly selected as the test images. To ensure objective as well as effective evaluation, the ground truth data are carefully generated on the click-through. Specifically, for each test image, the set of images clicked by the same query in the training set are taken as the semantically similar images. In addition, for training queries that share more than one common noun phrase with the query of the test image, their clicked images are also regarded as the similar ones. The other images in the training set are all used as dissimilar ones to the test image.

## 5.2 Protocols and Baseline Methods

We follow three evaluation protocols, i.e., hash lookup, recall and mean average precision (MAP), which are widely used in [4][13][26]. Hash lookup takes constant search time over a lookup table. We carry out hash lookup within a Hamming radius 2 of the query and report the search precision. Following [26], query failing in finding any hash bucket receives zero score in precision. Given the number of the retrieved images, the fraction of relevant ones, i.e., recall, is given as well. MAP is further exploited to evaluate the ranking quality in Hamming space for each query image.

We compare the following approaches for performance evaluation:

- Locality Sensitive Hashing [3] (*LSH*). *LSH* aims to map similar examples to the same bucket with high probability by using a Gaussian random projection matrix. The property of locality in the original space will be largely preserved in the Hamming space.
- PCA Hashing (*PCAH*). *PCAH* simply uses the matrix of top- $k$  PCA directions as projection matrix.
- Spectral Hashing [30] (*SH*). *SH* is based on quantizing the values of analytical eigenfunctions computed along PCA directions of the data.
- Semi-supervised Hashing [26] (*SSH*). *SSH* formulates the hashing problem as minimizing empirical error on the labeled data while maximizing variance of hash bits over both the labeled and unlabeled data.
- Semi-supervised Hashing with Semantic Confidence (*SHSC*) based on our proposal presented in Section 4.1.
- Ranking SHSC (*RSHSC*) based on Algorithm 1.
- Kernel-based Supervised Hashing [13] (*KSH*). *KSH* employs a kernel formulation for learning the hash functions to handle linearly inseparable data. We name this run as *KSH* in short.
- Kernel-based Ranking SHSC (*KRSHSC*) based on our proposal in Section 4.3. A slightly different of this run is name as *KRSHSC*<sup>-</sup>, which measures the quantization loss only on the labeled data.

## 5.3 Parameter Settings

To ensure that all the methods are comparable under the same setting, as in [13], we use the same Gaussian RBF kernel  $\kappa(\mathbf{v}_i, \mathbf{v}_j) = \exp(-\|\mathbf{v}_i - \mathbf{v}_j\|/2\sigma^2)$  and  $m = 300$  support samples in all kernel-based methods. The parameter  $\sigma$  is tuned on each dataset. The parameters  $\alpha$  and  $\beta$  are selected from  $\{0.2, 0.4, 0.6, 0.8, 1.0\}$  and the optimal values are determined by using a validation set. Finally,  $\alpha$  and  $\beta$  are both set to 0.8.

## 5.4 Results on CIFAR-10 Dataset

**Performance Comparison.** Figure 5(a) shows the MAP performances of nine runs on CIFAR-10 dataset. Overall, the results on MAP across different lengths of hash code consistently indicate that hashing with semantic confidence leads to a performance boost. There is a significant performance gap between the kernel-based and linear runs. It is not very surprising to see that *LSH* provides the worst MAP performance since the random hash functions lack discrimination power for small bit lengths. On the other hand, *PCAH* and *SH* work relatively well for small bit sizes, but getting worse as the number of bits increases, indicating that PCA is effective in preserving semantic consistency for small hash code lengths. Furthermore, by additionally incorporating semantic confidence, *SHSC* exhibits better performance than *SSH*. Similar is spirit, *KRSHSC*<sup>-</sup> improves *KSH*, but the performance is still lower than that of *KRSHSC*, demonstrating the advantage of exploiting the unlabeled data in semi-supervised hashing approaches. The improvement can also be observed of *RSHSC* compared to *SHSC*.

In the evaluation of hash lookup within Hamming radius 2 as shown in Figure 5(b), the precisions for most of the

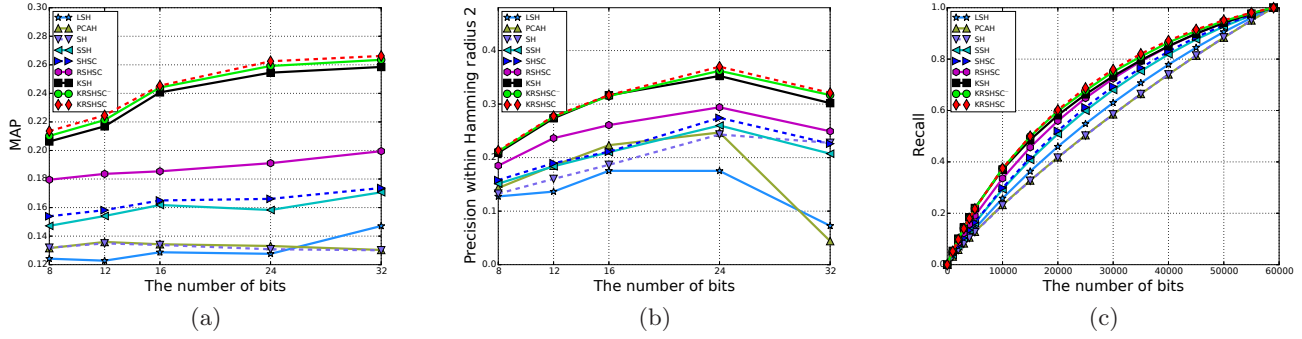


Figure 5: Comparisons with state-of-the-art approaches on CIFAR-10 dataset. (a) Mean average precision (MAP) performance. (b) Precision within Hamming radius 2 using hash lookup. (c) Recall curves with 32 bits. For better viewing, please see original color pdf file.

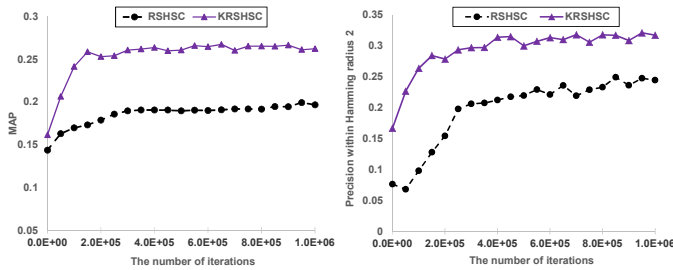


Figure 6: MAP performances and Precisions within Hamming radius 2 using hash lookup with the increase of the number of iterations on CIFAR-10 dataset with 32 bits.

compared methods drop when a longer size of hash code is used (32 bits in our case). This is because the number of samples falling into a bucket decreases exponentially for longer sizes of hash code. Therefore, for some query images, there are even no any neighbor in a Hamming ball of radius 2. Even in this case, our proposed *KRSHSC* provides the best performance and the drop in precision for long size of hash code is less than others.

We further details the recall at different numbers of returned examples in Figure 5(c). The results confirm the trends seen in Figure 5(a) and demonstrate performance improvement using the proposed semi-supervised hashing with semantic confidence approaches, especially *KRSHSC*, over other runs. In addition, to verify that the performance of different approaches is not by chance, we conducted significance test using the randomization test [22]. The number of iterations used in the randomization is 100,000 and at 0.05 significance level. *KRSHSC* is found to be significantly better than others.

**Varying the number of iterations.** In our *RSHSC* and *KRSHSC* algorithms, the learning of hash function is an iterative process. Next, we conducted experiments to evaluate the performances of our proposed approaches by varying the number of iterations from 1 to 1.0 million on CIFAR-10 dataset. Note that the training time grows linearly with the number of iterations.

MAP performances and precisions within Hamming radius 2 hash lookup with 32 hashing bits are reported in

Figure 6. Not surprisingly, we can observe that the performances of *RSHSC* and *KRSHSC* are both improved with the increase of iterations. Furthermore, after a number of iterations (600K in our case), the performances of *RSHSC* and *KRSHSC* change very smoothly, as the algorithms have already received sufficient knowledge to learn a good hash function.

## 5.5 Results on Clicktute Dataset

Figure 7 shows the experimental results on Clicktute dataset. As for some queries in Clicktute dataset, there are only tens of or even less than ten clicked (relevant) images, making the search task very challenging. MAP performance and precision with Hamming radius 2 using hash lookup are given in Figure 7(a) and (b), respectively. Our *KRSHSC* approach consistently outperforms other runs. In particular, the MAP performance and precision with Hamming radius 2 using hash lookup of *KRSHSC* can achieve 0.0818 and 0.1735 with 48 hash bits, which make the improvement over the best competitor *KSH* by 3.5% and 12.8%. Methods that learn hash functions with semantic confidence, e.g. *KRSHSC*<sup>-</sup> and *SHSC*, are generally better than *KSH* and *SSH*, respectively. Similar to the observations on CIFAR-10 dataset, *LSH* performs poorly especially for small bit sizes and *SH* leads to better performance gain than *PCAH* for longer hash code. Figure 8 showcases some exemplar query images and their retrieved neighbors with 48 bits. *KRSHSC* still exhibits the best search quality in terms of visual relevance.

## 5.6 Complexity Analysis

The time complexities for training *RSHSC* and *KRSHSC* are  $T_{max} \times \mathcal{O}(nKD + DK^2 + D)$  and  $T_{max} \times \mathcal{O}(nKm + mK^2 + m)$ , respectively, which scales linearly with  $n$  given  $n \gg D > m > K$ . In practice, take the training on 1 million triplets for example, *KRSHSC* takes about 30 minutes on a server with 2.40GHz CPU and 128GB RAM. For each query, the hashing time of *RSHSC* and *KRSHSC* are  $\mathcal{O}(KD)$  and  $\mathcal{O}(Dm + Km)$ , respectively.

## 6. DISCUSSION AND CONCLUSION

In this paper, we have presented an important concept, i.e. semantic confidence, for the learning of hash function. Particularly, we propose two ways of measuring the



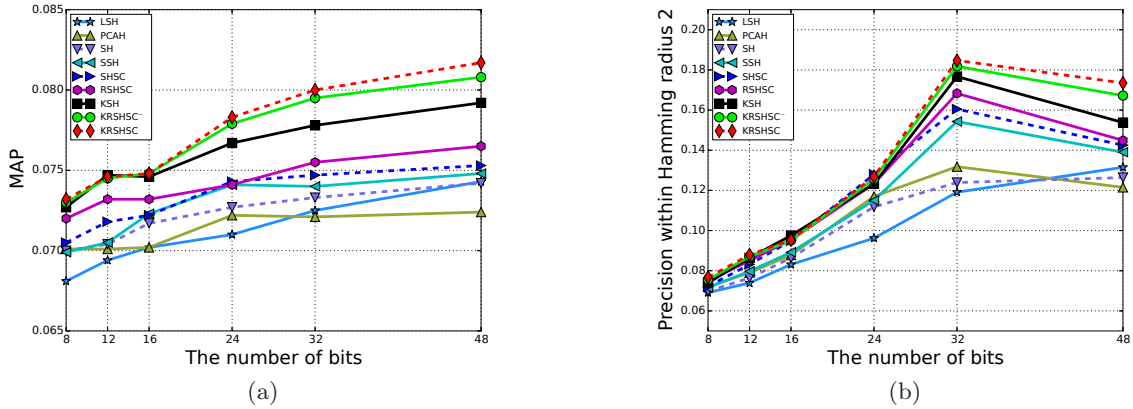


Figure 7: Comparisons with state-of-the-art approaches on Clicktute dataset (Better viewed in color). (a) Mean average precision (MAP) performance. (b) Precision within Hamming radius 2 using hash lookup.

semantic confidence, by neighbor voting and click counts, where the former is for general purpose and the latter exploits the click-through data which is largely available by search engine. With the semantic confidence, pairwise and triplet relationships are deployed and further incorporated into semi-supervised hashing learning framework with pairwise and listwise supervision, respectively. Finally, a kernel-based version is proposed to handle the linearly inseparable data.

We performed extensive experiments on two image datasets and compared with the state-of-the-art hashing techniques. Experimental results demonstrated that the proposed semi-supervised hashing with semantic confidence yields superior performance. The current work can be extended with the design of multiple listwise supervised hash tables, which is expected to show even better performance.

## Acknowledgments

This work was supported in part by the 973 Programme under Grant 2015CB351803, the 863 Programme under Grant 2014AA015102, and National Natural Science Foundation of China (No. 61272290, No. 61325009).

## 7. REFERENCES

- [1] C. F. Cadieu, H. Hong, D. Yamins, N. Pinto, N. J. Majaj, and J. J. DiCarlo. The neural representation benchmark and its evaluation on brain and machine. In *ICLR*, 2013.
- [2] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *arXiv preprint arXiv:1310.1531*, 2013.
- [3] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *VLDB*, 1999.
- [4] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Trans. PAMI*, 35(12):2916–2929, 2013.
- [5] R. Herbrich, T. Graepel, and K. Obermayer. *Large Margin Rank Boundaries for Ordinal Regression*. MIT Press, January 2000.
- [6] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.
- [7] X.-S. Hua, L. Yang, J. Wang, J. Wang, M. Ye, K. Wang, Y. Rui, and J. Li. Clickage: Towards bridging semantic and intent gaps via mining click logs of search engines. In *ACM MM*, 2013.
- [8] S. Kim and S. Choi. Semi-supervised discriminant hashing. In *ICDM*, 2011.
- [9] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.
- [10] B. Kulis and T. Darrell. Learning to hash with binary reconstructive embeddings. In *NIPS*, 2009.
- [11] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing. *IEEE Trans. PAMI*, 34(6):1092–1104, 2012.
- [12] W. Liu, C. Mu, S. Kumar, and S.-F. Chang. Discrete graph hashing. In *NIPS*, 2014.
- [13] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *CVPR*, 2012.
- [14] W. Liu, J. Wang, S. Kumar, and S.-F. Chang. Hashing with graphs. In *ICML*, 2011.
- [15] Y. Mu, J. Shen, and S. Yan. Weakly-supervised hashing in kernel space. In *CVPR*, 2010.
- [16] M. Norouzi and D. M. Blei. Minimal loss hashing for compact binary codes. In *ICML*, 2011.
- [17] M. Norouzi, D. M. Blei, and R. Salakhutdinov. Hamming distance metric learning. In *NIPS*, 2012.
- [18] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV*, 42(3):145–175, 2001.
- [19] Y. Pan, T. Yao, T. Mei, H. Li, C. W. Ngo, and Y. Rui. Click-through-based cross-view learning for image search. In *SIGIR*, 2014.
- [20] P. Ram, D. Lee, H. Ouyang, and A. G. Gray. Rank-approximate nearest neighbor search: Retaining meaning and speed in high dimensions. In *NIPS*, 2009.
- [21] M. Rastegari, J. Choi, S. Fakhraei, D. Hal, and L. Davis. Predictable dual-view hashing. In *ICML*, 2013.

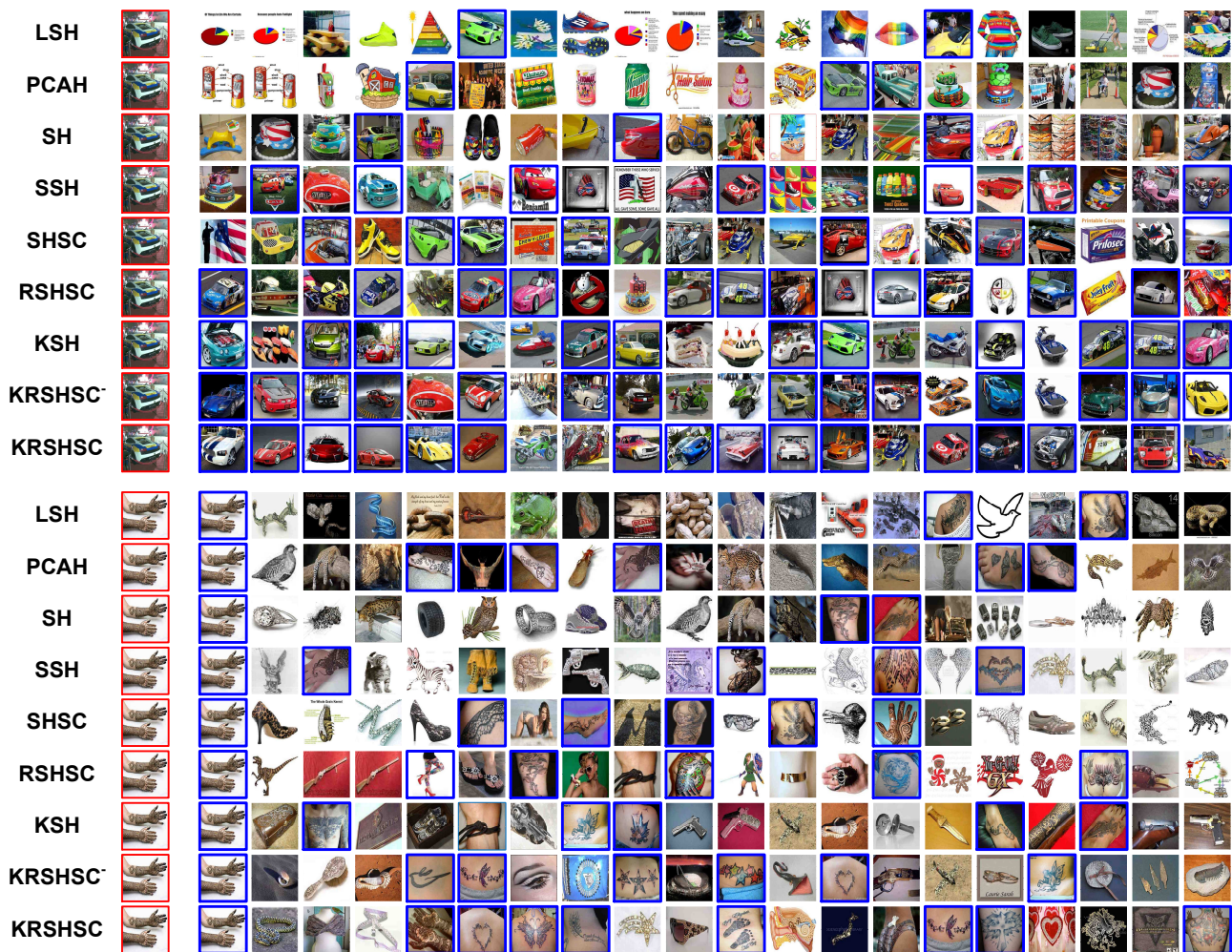


Figure 8: Examples showing the top 20 image search results by different methods in response to two query images on Clicktute dataset (better viewed in color). In each row, the first image with a red bounding box is the query image and the relevant images in the retrieved list are enclosed in a blue bounding box.

- [22] J. P. Romano. On the behavior of randomization tests without a group invariance assumption. *Journal of the American Statistical Association*, 85(411):686–692, 1990.
- [23] R. Salakhutdinov and G. Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2006.
- [24] C. Strecha, A. M. Bronstein, M. M. Bronstein, and P. Fua. Ldhash: Improved matching with smaller descriptors. *IEEE Trans. PAMI*, 34(1):66–78, 2012.
- [25] A. Torralba, R. Fergus, and W. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Trans. PAMI*, 30(11):1958–1970, 2008.
- [26] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for large-scale search. *IEEE Trans. PAMI*, 34(12):2393–2406, 2012.
- [27] J. Wang, W. Liu, A. X. Sun, and Y.-G. Jiang. Learning hash codes with listwise supervision. In *ICCV*, 2013.
- [28] Q. Wang, L. Si, and D. Zhang. Learning to hash with partial tags: Exploring correlation between tags and hashing bits for large scale image retrieval. In *ECCV*, 2014.
- [29] Q. Wang, D. Zhang, and L. Si. Semantic hashing using tags and topic modeling. In *SIGIR*, 2013.
- [30] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, 2008.
- [31] H. Xia, P. Wu, S. C. Hoi, and R. Jin. Boosting multi-kernel locality-sensitive hashing for scalable image retrieval. In *SIGIR*, 2012.
- [32] T. Yao, T. Mei, C.-W. Ngo, and S. Li. Annotation for free: Video tagging by mining user search behavior. In *ACM MM*, 2013.
- [33] L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. In *NIPS*, 2004.
- [34] X. Zhu, W. Nejdl, and M. Georgescu. An adaptive teleportation random walk model for learning social tag relevance. In *SIGIR*, 2014.