Semantics-Preserving Hashing for Cross-View Retrieval

Zijia Lin[†,‡] Guiguang Ding[‡] Mingqing Hu[§] Jianmin Wang[‡]
[†]Department of Computer Science and Technology, Tsinghua University, Beijing, P.R.China
[‡]School of Software, Tsinghua University, Beijing, P.R.China
[§]Institute of Computing Technology, Chinese Academy of Sciences, Beijing, P.R.China

Abstract

With benefits of low storage costs and high query speeds, hashing methods are widely researched for efficiently retrieving large-scale data, which commonly contains multiple views, e.g. *a news report with images, videos and texts. In this paper, we study the problem of cross-view retrieval and propose an effective* ***Se****mantics-****P****reserving* ***H****ashing method, termed SePH. Given semantic affinities of training data as supervised information, SePH transforms them into a probability distribution and approximates it with to-be-learnt hash codes in Hamming space via minimizing the Kullback-Leibler divergence. Then kernel logistic regression with a sampling strategy is utilized to learn the non-linear projections from features in each view to the learnt hash codes. And for any unseen instance, predicted hash codes and their corresponding output probabilities from observed views are utilized to determine its unified hash code, using a novel probabilistic approach. Extensive experiments conducted on three benchmark datasets well demonstrate the effectiveness and reasonableness of SePH.*

1. Introduction

For numerous algorithms in the field of information retrieval, computer vision, *etc*., retrieving nearest neighbours for an instance plays a fundamental role. However, with the explosion of data recently, nearest neighbour retrieval from large-scale databases becomes challenging. To tackle that, hashing methods are proposed. By representing each instance as a k-bit binary hash code (*i.e*. 0 or 1), data can be compactly stored and Hamming distances can be fast calculated with bit-wise XOR operations. With such benefits of low storage costs and high query speeds, hashing methods attract much attention from both academia and industries.

Existing hashing methods can be roughly classified into two categories: single-view hashing [16, 21, 4, 11, 6, 13, 8, 22, 10] and multi-view hashing [1, 9, 20, 24, 23, 18, 27, 14, 3, 25, 17, 26, 19], with the former focusing on data with a single view while the latter focusing on that with multiple views, *e.g*. a news report with images, videos and texts. In this paper, we study the problem of cross-view retrieval for data with multiple views. Actually, as revealed in [19], cross-view retrieval is becoming popular, since it just needs one view of a query and can retrieve nearest neighbours in different views. For example, we can use a query image to retrieve relevant videos or texts from a database. Recently, various hashing methods for cross-view retrieval have been proposed, including unsupervised ones [27, 14, 3, 25] and supervised ones [1, 9, 24, 23, 18, 17, 26, 19].

In this paper, we propose a supervised **Se**mantics-**P**reserving **H**ashing method termed SePH for cross-view retrieval. Given the semantic affinities of training data as the supervised information, the proposed SePH firstly transforms them into a probability distribution $\mathcal{P}$ and approximates it in Hamming space, via transforming all pairwise Hamming distances between to-be-learnt hash codes into another probability distribution $\mathcal{Q}$ and minimizing its Kullback-Leibler divergence (abbreviated as KL-divergence) from $\mathcal{P}$. Unlike previous work [9, 26, 19] that utilizes the supervised information for *independently* weighting each pairwise distance/similarity between hash codes, SePH standardizes all Hamming distances by transforming each into a probability that *depends* on all others. And thus correlations between Hamming distances are incorporated for forcing the to-be-learnt hash codes to better preserve the semantic structure of training instances. With learnt semantics-preserving hash codes on training data, SePH utilizes kernel logistic regression with a sampling strategy as basic hash functions to model the projections from features in each view to the hash codes. And for any unseen instance, with predicted hash codes as well as their corresponding output probabilities from different observed views, SePH determines its unified hash code using a novel probabilistic approach. Note that SePH is a two-step hashing framework, and the learning of hash functions for out-of-sample extension is open for any effective method. Extensive experiments conducted on three benchmark datasets well demonstrate that SePH is effective and reasonable.

Contributions of our work can be summarized as follows.

- We propose an effective supervised cross-view hashing method termed SePH, which transforms the semantic affinities of training data into a probability distribution and approximates it with to-be-learnt hash codes in Hamming space via minimizing the KL-divergence.
- We propose a novel probabilistic approach for determining the unified hash code of any unseen instance, using predicted hash codes as well as the corresponding output probabilities from different observed views.

The remainder of this paper is organized as follows. Section 2 gives an overview of related work. Section 3 presents formula details of the proposed SePH. Then detailed descriptions of experiments are given in Section 4. And finally we draw conclusions in Section 5.

2. Related Work

Recently, various hashing methods have been proposed for cross-view retrieval, including unsupervised ones and supervised ones.

Unsupervised hashing methods [27, 14, 3, 25] generally focus on exploiting the intra-view and inter-view relations of training data with only features in different views to learn the projections from features to hash codes. J. Song *et al*. [14] proposed inter-media hashing (IMH) that introduces inter-view and intra-view consistency to learn linear hash functions for mapping features in different views into a common Hamming space. G. Ding *et al*. [3] proposed CMFH that learns unified hash codes of instances by collective matrix factorization with latent factor model from different views. J. Zhou *et al*. [25] proposed Latent Semantic Sparse Hashing (LSSH) that learns latent semantic features for images and texts respectively with sparse coding and matrix factorization, and then maps them to a joint abstraction space for generating unified hash codes. Though without supervised information, experiments showed that CMFH and LSSH could well exploit the latent semantic affinities of training data and yield state-of-the-art performance for cross-view retrieval.

Different from unsupervised cross-view hashing methods, supervised ones [1, 9, 24, 23, 18, 17, 26, 19] are proposed to further exploit available supervised information like labels or semantic affinities of training data for improving performance. M. Bronstein *et al*. [1] proposed CMSSH that models the projections from features in each view to hash codes as binary classification problems, and learns them with boosting algorithms. S. Kumar and R. Udupa [9] extended the single-view spectral hashing [16] to the case of multiple views and proposed a novel method termed CVH, which learns hash functions via minimizing the similarity-weighted Hamming distances between hash codes of training data. Y. Zhen and D. Yeung [23] proposed co-regularized hashing (CRH) for multi-view data, which learns hash functions for each bit of the hash codes by solving DC (difference of convex functions) programs and performs the learning for multiple bits via a boosting procedure. J. Zhou *et al*. [26] proposed KSH-CV to learn kernel hash functions via preserving inter-view similarities under an Adaboost framework. D. Zhang and W. Li [19] proposed SCM to integrate semantic labels into the hashing learning procedure via maximizing semantic correlations, which outperforms several state-of-the-art methods.

The proposed SePH in this paper is a supervised hashing method. And different from previous work mentioned above, when given semantic affinities of training data as the supervised information, SePH transforms them into a probability distribution $\mathcal{P}$ and approximates it in Hamming space, via transforming all pairwise Hamming distances between to-be-learnt hash codes into another probability distribution $\mathcal{Q}$ and minimizing its KL-divergence from $\mathcal{P}$. In that way, SePH standardizes all Hamming distances by transforming each into a probability that depends on all others, and thus incorporates the correlations between Hamming distances to make the to-be-learnt hash codes better preserve the semantic structure of training instances.

3. Proposed SePH

An illustration of the framework of the proposed SePH is given in Fig. 1. Like [3, 25], for each instance, SePH learns one unified hash code, instead of learning different hash codes respectively for each observed view as other previous work [1, 9, 26, 19]. And thus SePH can further reduce the storage costs. As illustrated in Fig. 1, given training data with different views and its corresponding affinity matrix indicating the semantic affinities between training instances, SePH firstly learns the semantics-preserving hash codes of training data via minimizing the KL-divergence of the derived probability distribution $\mathcal{Q}$ in Hamming space from that in semantic space (*i.e*. $\mathcal{P}$). Then hash functions for each view are learnt to project features into the learnt hash codes. And for any unseen instance, predicted hash codes as well as the corresponding output probabilities from learnt hash functions in all observed views are utilized to determine its unified hash code with a novel probabilistic approach. For ease of presentation, here we describe SePH with only two views. But it can be easily extended to cases with more views, as will be described later.

3.1. Problem Formulation

Assume that the training data consists of n instances, *i.e*. $\mathcal{O} = \{o_1, o_2, \ldots, o_n\}$ with o_i being the ith training instance, and their corresponding two views $\mathcal{X}$ and $\mathcal{Y}$ are observed. Here we denote $X_{n\times d_x}$ and $Y_{n\times d_y}$ as the view-specific feature matrices of the training data, which are built with feature vectors in respective views row by row. Then $X_{i,\cdot} \in \mathbb{R}^{d_x}$ and $Y_{i,\cdot} \in \mathbb{R}^{d_y}$, *i.e*. the ith rows in X and Y, are

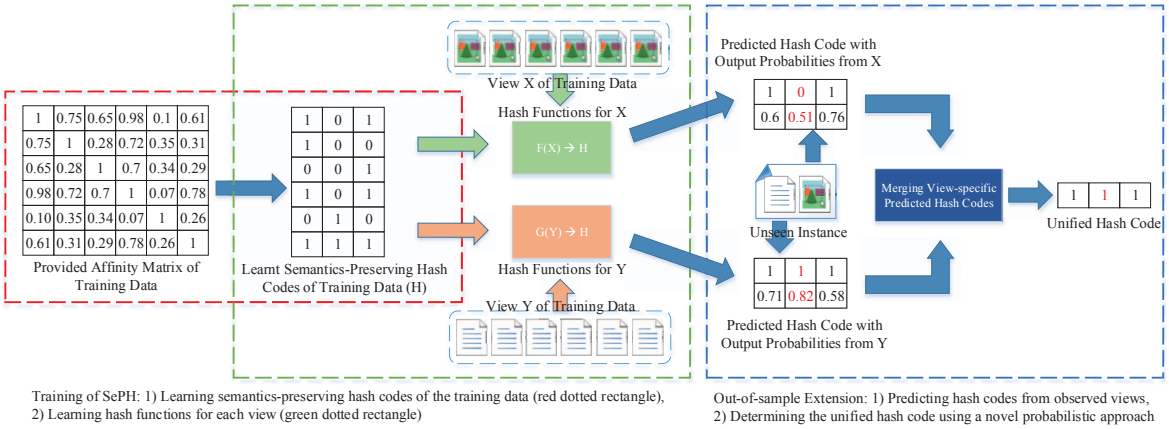

Figure 1. Framework of the proposed SePH, illustrated with two-view toy data.

respectively the feature vector of the ith training instance in $\mathcal{X}$ and $\mathcal{Y}$. The affinity matrix $A_{n\times n}$ of training data is also assumed to be provided, with $A_{i,j} \in [0, 1]$ indicating the semantic affinity between o_i and o_j. Here A is symmetric with $A_{i,j} = A_{j,i}$, and it provides supervised information for learning hash codes of training instances. Generally, A can be derived via manual scoring, or estimated from label vectors of training data, *e.g.* cosine similarities.

Given the training data with its affinity matrix A, the proposed SePH is to firstly learn semantics-preserving hash codes on it, and then learn hash functions for out-of-sample extension in each view, which can be seen as a two-step hashing framework similar to [21]. Here we denote $H_{n\times d_c}$ as the hash code matrix of training data, with its ith row $H_{i,\cdot} \in \{-1, 1\}^{d_c}$ being the d_c-bit hash code of o_i. Note that $\{-1, 1\}$ is utilized here for model simplicity, and it can be easily mapped into $\{0, 1\}$.

3.2. Semantics-Preserving Hashing

For a semantics-preserving hashing method, if o_i and o_j are semantically similar, their corresponding hash codes are supposed to be similar, and vice versa. Previous work on supervised cross-view hashing like [9, 26, 19] generally takes the provided semantic affinities to *independently* weight each pairwise distance/similarity between hash codes. Differently, SePH transforms the semantic affinities into a probability distribution $\mathcal{P}$ and approximates it in Hamming space, via transforming all pairwise Hamming distances between to-be-learnt hash codes into another probability distribution $\mathcal{Q}$ and minimizing its KL-divergence from $\mathcal{P}$. In that way, SePH standardizes all Hamming distances by transforming each into a probability *depending* on all others, and thus incorporates the correlations between Hamming distances to make the to-be-learnt hash codes better preserve the semantic structure of training instances.

Specifically, to derive $\mathcal{P}$, we define $p_{i,j}$ as the probability of observing the semantic similarity between o_i and o_j among all pairs of training instances. Furthermore, we assume $p_{i,j}$ to be proportional to the corresponding semantic affinity, *i.e.* $A_{i,j}$. With all probabilities forced to sum up to 1, *i.e.* $\sum_{i\neq j} p_{i,j} = 1$, we can derive $p_{i,j}$ as follows.

$$p_{i,j} = \frac{A_{i,j}}{\sum_{i\neq j} A_{i,j}} \tag{1}$$

To derive $\mathcal{Q}$, we define the probability of observing the similarity between o_i and o_j in Hamming space as $q_{i,j}$. Following t-SNE [15], we utilize a Student t-distribution with one degree of freedom to transform Hamming distances into probabilities, as shown in the following formula.

$$q_{i,j} = \frac{(1 + h(H_{i,\cdot}, H_{j,\cdot}))^{-1}}{\sum_{k\neq m}(1 + h(H_{k,\cdot}, H_{m,\cdot}))^{-1}} \tag{2}$$

where $h(\cdot, \cdot)$ denotes the Hamming distance between hash codes. With $H_{i,j} \in \{-1, 1\}$ for any i and j, the Hamming distance between hash codes of two instances can be derived from their squared Euclidean distance as formula (3).

$$h(H_{i,\cdot}, H_{j,\cdot}) = \frac{1}{4}\|H_{i,\cdot} - H_{j,\cdot}\|_2^2 \tag{3}$$

And $q_{i,j}$ can be further rewritten as follows, which is more tractable for optimization.

$$q_{i,j} = \frac{(1 + \frac{1}{4}\|H_{i,\cdot} - H_{j,\cdot}\|_2^2)^{-1}}{\sum_{k\neq m}(1 + \frac{1}{4}\|H_{k,\cdot} - H_{m,\cdot}\|_2^2)^{-1}} \tag{4}$$

Then the objective of SePH is to learn an optimal H that can make $\mathcal{Q}$ match $\mathcal{P}$ as well as possible, and thus preserve the semantic structure represented by $\mathcal{P}$. A natural measure for the difference between $\mathcal{Q}$ and $\mathcal{P}$ is the Kullback-Leibler divergence, as defined in the following formula.

$$D_{KL}(P\|Q) = \sum_{i\neq j} p_{i,j} \log \frac{p_{i,j}}{q_{i,j}} \tag{5}$$

And thus SePH minimizes $D_{KL}(P\|Q)$ to learn the optimal hash codes of training data, *i.e.* H, with its objective function given as follows.

$$\Psi_0 = \min_{H\in\{-1,1\}^{n\times d_c}} \sum_{i\neq j} p_{i,j}\log\frac{p_{i,j}}{q_{i,j}} \tag{6}$$

where $p_{i,j}$ and $q_{i,j}$ are respectively defined as formula (1) and (4). However, the objective function above is NP-hard for directly optimizing the optimal binary H. To make it tractable, we relax H to be real-valued, which is denoted as $\hat{H}$. And we introduce a regularizer to force the entries of $\hat{H}$ near to -1 or 1, thus making the learnt $\hat{H}$ near to the optimal binary H, as shown in the following formula.

$$\begin{aligned} \Psi &= \min_{\hat{H}\in\mathbb{R}^{n\times d_c}} \sum_{i\neq j} p_{i,j}\log\frac{p_{i,j}}{q_{i,j}} + \frac{\alpha}{C}\||\hat{H}| - \mathbf{I}\|_2^2 \\ s.t. \quad q_{i,j} &= \frac{(1+\frac{1}{4}\|\hat{H}_{i,\cdot}-\hat{H}_{j,\cdot}\|_2^2)^{-1}}{\sum_{k\neq m}(1+\frac{1}{4}\|\hat{H}_{k,\cdot}-\hat{H}_{m,\cdot}\|_2^2)^{-1}} \end{aligned} \tag{7}$$

where $\mathbf{I}$ is a matrix with all entries being 1, and $\||\hat{H}| - \mathbf{I}\|_2^2$ measures the quantization loss from real-valued $\hat{H}$ to binary H. Here α is a model parameter for balancing the KL-divergence and the quantization loss, and $C = n \times d_c$ is a normalizing factor to make α less dependent on the training set size and the hash code length.

3.3. Solution and Implementation Issues

The objective function Ψ of SePH is non-convex, meaning that we can only derive a local optimum. As Ψ is an unconstrained optimization problem, to learn a locally optimal $\hat{H}$, here we propose to utilize gradient descent based optimization methods. Specifically, the gradient w.r.t. $\hat{H}_{i,\cdot}$, *i.e.* the ith row of $\hat{H}$, can be derived as follows.

$$\begin{aligned} \frac{\partial\Psi}{\partial\hat{H}_{i,\cdot}} &= \sum_{j\neq i}(p_{i,j}-q_{i,j})(1+\frac{1}{4}\|\hat{H}_{i,\cdot}-\hat{H}_{j,\cdot}\|_2^2)^{-1}(\hat{H}_{i,\cdot}-\hat{H}_{j,\cdot}) \\ &\quad + \frac{2\alpha}{C}(|\hat{H}_{i,\cdot}| - \mathbf{1}^T)\odot\sigma(\hat{H}_{i,\cdot}) \end{aligned} \tag{8}$$

where $\odot$ denotes entry-wise multiplication, $\mathbf{1}$ is a d_c-dimensional column vector with all entries being 1, and $\sigma(\hat{H}_{i,\cdot})$ is a d_c-dimensional row vector consisting of the signs of all entries in $\hat{H}_{i,\cdot}$. Note that $\sigma(\hat{H}_{i,\cdot})$ is actually $\frac{\partial|\hat{H}_{i,\cdot}|}{\partial\hat{H}_{i,\cdot}}$, and for simplicity, the corresponding gradients w.r.t. zero entries are assigned as 0.

With gradients calculated as formula (8), effective gradient descent based optimization methods can be further applied to derive an optimal $\hat{H}$. And the corresponding binary hash code matrix H can be derived as the signs of entries in $\hat{H}$, *i.e.* $H = sign(\hat{H})$, with signs of zero entries set as 1. Then for out-of-sample extension, hash functions are learnt for each view to project features into hash codes, using H and the corresponding feature matrix (*i.e.* X or Y).

3.4. Learning Hash Functions

After deriving the optimal hash code matrix H, the learning of hash functions is open for any effective predictive models, like linear regression, SVM, logistic regression, *etc*. Actually, as [1] revealed, for any bit of the hash code, learning the corresponding hash function to project features into it can be modelled as a binary classification problem.

In this paper, we propose to utilize kernel logistic regression to learn the projections from features to hash codes for each view. The reason is two-fold: 1) kernel tricks is capable of modelling non-linear projections, 2) logistic regression can naturally provide probabilities with the predicted results. As will be described later, the output probabilities can be utilized to determine the unified hash code of an unseen instance. Kernel logistic regression is learnt independently in different views. For ease of presentation, here we only describe the learning process in view $\mathcal{X}$, which can be easily extended to other views.

Specifically, in kernel logistic regression, each feature vector $X_{i,\cdot}$ is mapped to the Reproducing Kernel Hilbert Space (RKHS) as $\phi(X_{i,\cdot})$, which also forms a kernel feature matrix Φ row by row. In RKHS, the inner product $\phi(X_{i,\cdot})\phi^T(X_{j,\cdot})$ between kernel features $\phi(X_{i,\cdot})$ and $\phi(X_{j,\cdot})$ can be efficiently calculated as $\kappa(X_{i,\cdot}, X_{j,\cdot})$, where $\kappa(\cdot,\cdot)$ is the introduced kernel function. Using non-linear kernel functions, linear projections from kernel features to hash codes essentially represent non-linear projections from original features to hash codes. For the kth bit of hash code ($1 \leq k \leq d_c$), denoting the linear projection in RKHS as $\mathbf{w}^{(k)}$ and values in the bit as $\mathbf{h}^{(k)} \in \{-1,1\}^{n\times 1}$, the objective function Θ of kernel logistic regression is as follows.

$$\Theta = \min_{\mathbf{w}^{(k)}} \sum_{i=1}^{n} \log(1+e^{-\mathbf{h}_i^{(k)}\phi(X_{i,\cdot})\mathbf{w}^{(k)}}) + \lambda\|\mathbf{w}^{(k)}\|_2^2 \tag{9}$$

where $\mathbf{h}_i^{(k)} \in \{-1,1\}$ is the ith entry in $\mathbf{h}^{(k)}$, and λ is a weighting parameter. Like kernel CCA [5], we require $\mathbf{w}^{(k)}$ to be in the span of the training kernel features, *i.e.* $\mathbf{w}^{(k)} = \Phi^T\mathbf{v}^{(k)}$ with $\mathbf{v}^{(k)}$ being the spanning weights. Then in formula (9), $\phi(X_{i,\cdot})\mathbf{w}^{(k)} = (\phi(X_{i,\cdot})\Phi^T)\mathbf{v}^{(k)}$. It can be seen that the training and predicting costs of kernel logistic regression are proportional to the training set size n, which may be unsuitable for large training sets.

Considering that training kernel features could be redundant for spanning $\mathbf{w}^{(k)}$, we propose to sample a fraction of them for building a much smaller kernel feature matrix $\hat{\Phi}$, and use it to span $\mathbf{w}^{(k)}$, *i.e.* $\mathbf{w}^{(k)} = \hat{\Phi}^T\hat{\mathbf{v}}^{(k)}$ where $\hat{\mathbf{v}}^{(k)}$ is the to-be-learnt s-dimensional weighting vector with $s(s \ll n)$ being the sampling size. Then the training and predicting costs of kernel logistic regression will depend on the sampling size rather than the training set size, making it more scalable for training and more efficient for predicting. As for sampling, one can utilize random sampling or other

strategies like k-means.

In view $\mathcal{X}$, by optimizing Θ for each bit of the hash code, its corresponding $\hat{\mathbf{v}}^{(k)}$ can be derived, and then $\{\hat{\Phi}, \hat{\mathbf{v}}^{(1)}, \hat{\mathbf{v}}^{(2)}, \ldots, \hat{\mathbf{v}}^{(d_c)}\}$ forms the hash function set f_X. Note that here $\hat{\Phi}$ is shared by all $\hat{\mathbf{v}}^{(k)}(1 \leq k \leq d_c)$, and thus the time costs of training and predicting for all d_c kernel logistic regressions can be further reduced.

3.5. Generating Hash Codes

With kernel logistic regression learnt in each view, given an unseen instance, predicted hash codes with their corresponding probabilities from observed views will be output. For example, in view $\mathcal{X}$, assuming that the feature vector of an unseen instance is $\mathbf{x}$, the kth bit of its view-specific predicted hash code $\mathbf{c}^{\mathcal{X}}$, denoted as $\mathbf{c}_k^{\mathcal{X}}$, is determined as the sign of $(\phi(\mathbf{x})\hat{\Phi}^T)\hat{\mathbf{v}}^{(k)}$. With $b \in \{-1, 1\}$, the corresponding output probabilities can be calculated as follows.

$$p(\mathbf{c}_k^{\mathcal{X}} = b|\mathbf{x}) = \left(1 + e^{-b\left(\phi(\mathbf{x})\hat{\Phi}^T\right)\hat{\mathbf{v}}^{(k)}}\right)^{-1} \tag{10}$$

For any unseen instance with only one view observed, it is straightforward to determine its unified hash code as the view-specific predicted one. As for those with both views observed, predicted hash codes from different views need to be merged to determine their unified hash codes, especially when the predicted hash codes from different views are inconsistent, as illustrated in Fig. 1. Here we propose a novel probabilistic approach for determining each bit of the unified hash code of an unseen instance. Assume that $\mathbf{x}$ and $\mathbf{y}$ are respectively the feature vectors of an unseen instance in different observed views, and $\mathbf{c}_k \in \{-1, 1\}$ is the kth bit of its unified hash code $\mathbf{c}$. Then $\mathbf{c}_k$ is determined as follows.

$$\mathbf{c}_k = sign\Big(p(\mathbf{c}_k = 1|\mathbf{x}, \mathbf{y}) - p(\mathbf{c}_k = -1|\mathbf{x}, \mathbf{y})\Big) \tag{11}$$

By applying the Bayes' theorem and assuming that different views are conditionally independent on $\mathbf{c}_k$, we can further derive that

$$\begin{aligned}\mathbf{c}_k = sign\Big(&p(\mathbf{x}|\mathbf{c}_k = 1)p(\mathbf{y}|\mathbf{c}_k = 1)p(\mathbf{c}_k = 1) \\ &-p(\mathbf{x}|\mathbf{c}_k = -1)p(\mathbf{y}|\mathbf{c}_k = -1)p(\mathbf{c}_k = -1)\Big)\end{aligned} \tag{12}$$

Considering the balance between -1 and 1, $p(\mathbf{c}_k = 1)$ and $p(\mathbf{c}_k = -1)$ are assumed to be equal. With further derivations applying the Bayes' theorem, we can finally get the following formula.

$$\begin{aligned}\mathbf{c}_k = sign\Big(&p(\mathbf{c}_k = 1|\mathbf{x})p(\mathbf{c}_k = 1|\mathbf{y}) \\ &-p(\mathbf{c}_k = -1|\mathbf{x})p(\mathbf{c}_k = -1|\mathbf{y})\Big)\end{aligned} \tag{13}$$

where $p(\mathbf{c}_k = b|\mathbf{z}) = p(\mathbf{c}_k^{\mathcal{Z}} = b|\mathbf{z})$ with $\mathbf{z} \in \{\mathbf{x}, \mathbf{y}\}, b \in \{-1, 1\}, \mathcal{Z} \in \{\mathcal{X}, \mathcal{Y}\}$, and all can be calculated as formula (10) by the corresponding learnt hash functions. Then by determining $\mathbf{c}_k$ for all $1 \leq k \leq d_c$, the unified hash code $\mathbf{c}$ of the unseen instance will be generated.

	Wiki	MIRFlickr	NUS-WIDE
Dataset Size	2,866	16,738	186,577
Training Set	2,173	5,000	5,000
Query Set	693	836	1,866
Nr. of Labels	10	24	10

Table 1. Statistics of three benchmark datasets.

3.6. Extensions

As mentioned previously, the proposed SePH can be easily extended to cases with more than two views. Actually, the training processes in those cases are nearly the same, except that hash functions for more views need to be learnt independently. As for out-of-sample extension, with similar derivations, formula (13) is extended as follows.

$$\mathbf{c}_k = sign\Big(\prod_{i=1}^{m} p(\mathbf{c}_k = 1|\mathbf{z}^i) - \prod_{i=1}^{m} p(\mathbf{c}_k = -1|\mathbf{z}^i)\Big) \tag{14}$$

where $m \geq 1$ is the number of observed views, $\mathbf{z}^i$ is the feature vector in the ith view, and all needed probabilities can also be calculated as formula (10).

4. Experiments

4.1. Experimental Settings

To validate the proposed SePH, we conduct experiments on three benchmark datasets, *i.e.* Wiki [12], MIRFlickr [7] and NUS-WIDE [2]. All datasets are with views of image and text. Some statistics of them are given in Table 1.

Wiki consists of 2,866 instances collected from Wikipedia. For each instance, the image view is represented as a 128-D Bag-of-Visual-Words SIFT feature vector and the text view as a 10-D topic vector. Each instance is annotated with one of 10 provided labels. Here we take 25% of the dataset as the query set and the rest as the retrieval set.

MIRFlickr originally contains 25,000 instances collected from Flickr, each being an image with its associated textual tags. Each instance is manually annotated with some of 24 provided unique labels. For pretreatment, we only keep those textual tags appearing at least 20 times, and then remove instances without textual tags or manually annotated labels. For each instance, the image view is represented with a 150-D edge histogram and the text view as a 500-D feature vector derived from PCA on its binary tagging vector w.r.t. the remaining textual tags. We take 5% of the dataset as the query set, and the rest as the retrieval set.

NUS-WIDE is a real-world web image dataset originally containing 269,648 instances, with each being an image with its associated textual tags. Each instance is manually annotated with at least one of 81 provided labels. Following [3, 25], only the top 10 most frequent labels and the corresponding 186,577 annotated instances are kept. For each

instance, the image view is represented as a 500-D Bag-of-Visual-Words SIFT feature vector and the text view as a binary tagging vector w.r.t. the top 1,000 most frequent tags. Here we take 1% of the dataset as the query set and the rest as the retrieval set.

Like [3], we take the retrieval set of Wiki as its training set due to its small size, while for the large MIRFlickr and NUS-WIDE, to reduce computational costs, we respectively sample 5,000 instances from their retrieval sets to form the training sets. Note that for SePH, the learnt hash codes of training instances are discarded after hash functions are learnt, and it utilizes the learnt hash functions to generate hash codes for all instances in the dataset.

In our experiments, the affinity matrix of each dataset, *i.e*. A in formula (1), is derived with the cosine similarities between labelling vectors of training instances w.r.t. the provided labels. And we empirically set the only model parameter α in the objective function of SePH (*i.e*. formula (7)) as 0.01 for all datasets. To learn kernel logistic regression in each view, λ in formula (9) is automatically set via cross-validation, and a RBF kernel with σ^2 set as the mean squared Euclidean distance between training features is utilized, with the sampling size set as 500 for all datasets. We perform both random sampling and k-means for kernel logistic regression in SePH, which are respectively denoted as SePH$_{rnd}$ and SePH$_{km}$.

SePH is compared with various state-of-the-art cross-view hashing methods. Specifically, we take the supervised CMSSH [1], CVH [9], KSH-CV [26], SCM-Orth and SCM-Seq [19], and the unsupervised IMH [14], LSSH [25] and CMFH [3] as baselines. It should be noticed that we utilize the provided labels of training instances to calculate the affinity matrices for IMH, and thus it is actually supervised in our experiments. Source codes of most baselines are kindly provided by the authors, while for CVH, we implement it as its codes are not available. Parameters of baselines are carefully tuned and here we report their best performance. Note that for SePH and baselines with a non-convex objective function, we perform 10 runs for each and take the average performance for comparison.

Performance of all cross-view hashing methods are measured with the widely-used *mean average precision* (mAP), as defined in formula (15).

$$mAP = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{m_i}\sum_{j=1}^{m_i} precision(R_{i,j}) \qquad (15)$$

where $|Q|$ is the size of the query set Q, m_i is the number of ground-truth relevant instances in the database for the ith query, $R_{i,j}$ is the subset of ranked retrieval results from the top one to the jth ground-truth relevant one, and thus $precision(R_{i,j})$ is the precision calculated in $R_{i,j}$. Following [3, 25], the ground-truth relevant instances for a query are defined as those sharing at least one label with it.

4.2. Experimental Results

For SePH and all baselines, we report their cross-view retrieval performance on all datasets in Table 2, including the performance of retrieving text with image (*i.e*. "Image Query v.s. Text Database") and that of retrieving image with text (*i.e*. "Text Query v.s. Image Database"). In both cases, we respectively take one view of query instances for generating their hash codes.

From the experimental results, we can draw the following observations. 1) On all benchmark datasets, the proposed SePH, *i.e*. SePH$_{rnd}$ and SePH$_{km}$, significantly outperforms all baselines with different hash code lengths, which well demonstrates its effectiveness. Particularly, in cases of "Text Query v.s. Image Database", SePH outperforms the best baselines by at least 15% and at most 41%. The superiority of SePH can be attributed to its capability to better preserve semantic affinities in Hamming space, as well as the effectiveness of kernel logistic regression to model the non-linear projections from features to hash codes for each view. 2) With the hash code length increasing, generally the performance of SePH keeps increasing, which reflects its capability of utilizing longer hash codes to better preserve semantic affinities. Meanwhile, performance of some baselines like CMSSH, KSH-CV and SCM-Orth decreases, which is also observed in previous work [26, 19, 3]. As analysed in [3], that can be due to the imbalance between bits in the hash codes learnt via eigenvalue or singular value decomposition. 3) Generally SePH$_{km}$ is superior to SePH$_{rnd}$, while the superiority is minor ($< 2\%$), meaning that the learning of kernel logistic regression is not sensitive to the sampling strategy. Hence for simplicity, one can utilize random sampling for real-world applications.

To validate the superiority of SePH in preserving semantic affinities, we further analyse the quality of the learnt hash codes on training sets. Specifically, we perform cross-view retrieval on a training set with the learnt hash codes, using one as a query and the rest as the retrieval set every time, and then measure the corresponding performance with mAP. As the ground-truth relevance between instances is defined with provided semantic labels, the mAP value can quantitatively reflect how well the learnt hash codes preserve the original semantic affinities. Fig. 2(a) and Fig. 2(b) illustrate the cross-view retrieval performance of SePH on the training set of the largest NUS-WIDE with different hash code lengths. Performance of baselines is also presented for comparison. It can be seen that SePH significantly outperforms the baselines, with the corresponding $mAP \geq 0.9$. And thus the hash codes learnt by SePH are high-quality, well preserving the semantic affinities of training instances. Similar results are observed on other datasets.

Since the objective function of SePH (*i.e*. formula (7)) is non-convex, here we further conduct experiments to analyse the effects of initial values, as they can result in different lo-

		Wiki				MIRFlickr				NUS-WIDE			
		16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits
Image Query *v.s.* Text Database	CMSSH [1]	0.1877	0.1771	0.1646	0.1552	0.5728	0.5743	0.5706	0.5706	0.4063	0.3927	0.3939	0.3739
	CVH [9]	0.1257	0.1212	0.1215	0.1171	0.6067	0.6177	0.6157	0.6074	0.3687	0.4182	0.4602	0.4466
	IMH [14]	0.1573	0.1575	0.1568	0.1651	0.6016	0.6120	0.6070	0.5982	0.4187	0.3975	0.3778	0.3668
	LSSH [25]	0.2141	0.2216	0.2218	0.2211	0.5784	0.5804	0.5797	0.5816	0.3900	0.3924	0.3962	0.3966
	CMFH [3]	0.2132	0.2259	0.2362	0.2419	0.5861	0.5835	0.5844	0.5849	0.4267	0.4229	0.4207	0.4182
	KSH-CV [26]	0.1965	0.1839	0.1701	0.1662	0.5793	0.5767	0.5732	0.5744	0.4229	0.4162	0.4026	0.3877
	SCM-Orth [19]	0.1598	0.1460	0.1383	0.1131	0.5854	0.5751	0.5704	0.5649	0.3787	0.3668	0.3593	0.3520
	SCM-Seq [19]	0.2210	0.2337	0.2442	0.2596	0.6237	0.6343	0.6448	0.6489	0.4842	0.4941	0.4947	0.4965
	SePH$_{rnd}$	0.2762	**0.2965**	0.3049	0.3131	0.6720	0.6761	**0.6794**	0.6814	0.5394	0.5454	0.5499	0.5556
	SePH$_{km}$	**0.2787**	0.2956	**0.3064**	**0.3134**	**0.6723**	**0.6771**	0.6783	**0.6817**	**0.5421**	**0.5499**	**0.5537**	**0.5601**
Text Query *v.s.* Image Database	CMSSH [1]	0.1630	0.1617	0.1539	0.1517	0.5715	0.5732	0.5699	0.5697	0.3874	0.3849	0.3704	0.3699
	CVH [9]	0.1185	0.1034	0.1024	0.0990	0.6026	0.6041	0.6017	0.5972	0.3646	0.4024	0.4339	0.4255
	IMH [14]	0.1463	0.1311	0.1290	0.1301	0.5895	0.6031	0.6010	0.5930	0.4053	0.3892	0.3758	0.3627
	LSSH [25]	0.5031	0.5224	0.5293	0.5346	0.5898	0.5927	0.5932	0.5932	0.4286	0.4248	0.4248	0.4175
	CMFH [3]	0.4884	0.5132	0.5269	0.5375	0.5937	0.5919	0.5931	0.5919	0.4627	0.4556	0.4518	0.4478
	KSH-CV [26]	0.1710	0.1665	0.1696	0.1576	0.5786	0.5763	0.5728	0.5715	0.4088	0.3906	0.3869	0.3834
	SCM-Orth [19]	0.1553	0.1389	0.1262	0.1096	0.5857	0.5747	0.5672	0.5604	0.3756	0.3641	0.3565	0.3523
	SCM-Seq [19]	0.2134	0.2366	0.2479	0.2573	0.6133	0.6209	0.6295	0.6340	0.4536	0.4620	0.4630	0.4644
	SePH$_{rnd}$	0.6312	**0.6581**	0.6637	0.6695	0.7178	0.7243	0.7287	0.7313	0.6230	0.6331	0.6407	0.6489
	SePH$_{km}$	**0.6318**	0.6577	**0.6646**	**0.6709**	**0.7197**	**0.7271**	**0.7309**	**0.7354**	**0.6302**	**0.6425**	**0.6506**	**0.6580**

Table 2. Cross-view retrieval performance of the proposed SePH (*i.e.* SePH$_{rnd}$ and SePH$_{km}$) and compared baselines on all benchmark datasets with different hash code lengths, in terms of mAP.

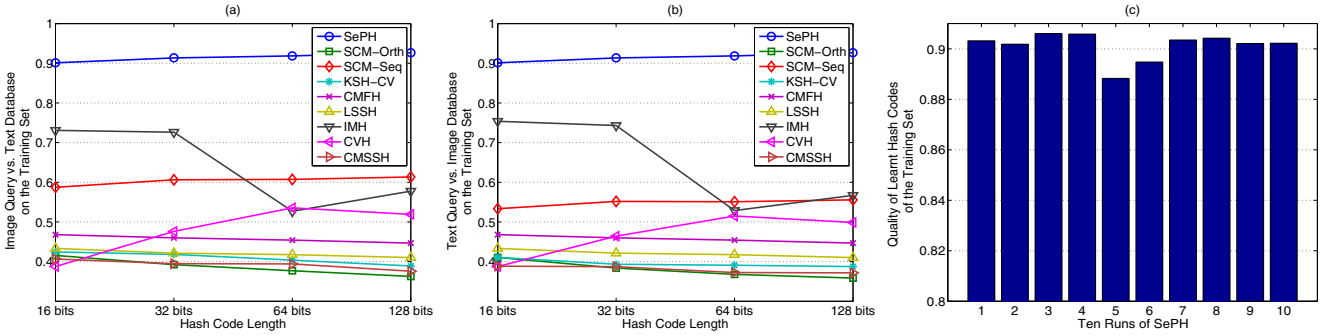

Figure 2. Inside analyses for the training processes of SePH and baselines on the largest NUS-WIDE. Sub-figure (a) and (b) illustrate the cross-view retrieval performance of all algorithms on the training set with different hash code lengths, which essentially reflects the quality of the hash codes generated by each algorithm for the training set. Sub-figure (c) shows the effects of initial values on the performance of SePH in 10 runs, with the hash code length fixed as 16 bits.

cal optimums for learning hash codes of training instances. Specifically, we fix the hash code length as 16 bits, and perform 10 runs for SePH with different initial values on the largest NUS-WIDE. Then quality of the learnt hash codes of the training set is measured with cross-view retrieval performance on it. Note that as SePH learns a unified hash code for each training instance, on the training set the performance of "Image Query v.s. Text Database" equals that of "Text Query v.s. Image Database", and thus we only show the former one in Fig. 2(c). It can be seen that the quality of learnt hash codes varies little using different initial values, with the corresponding standard error of mAP values being around 0.002. Similar results are also observed on other datasets and with other hash code lengths. And thus the performance of SePH is not sensitive to initial values.

4.3. Effects of Model Parameters

In previous experiments, we empirically set the only model parameter α in the objective function of SePH (*i.e.* formula (7)) as 0.01. Here we conduct experiments to analyse its effects on the quality of the learnt hash codes of a training set. Specifically, with the hash code length fixed as 16 bits, we vary α in $\{0, 10^{-4}, 10^{-3}, \ldots, 1\}$ for each dataset to learn hash codes of the corresponding training set, and then measure their quality with cross-view retrieval performance on the training set. Note that here the initial values for SePH are identical for each α, so as to remove their effects. Fig. 3(a) illustrates the effects of α on all three datasets. It can be seen that with α increasing, the quality of learnt hash codes for training sets of MIRFlickr and

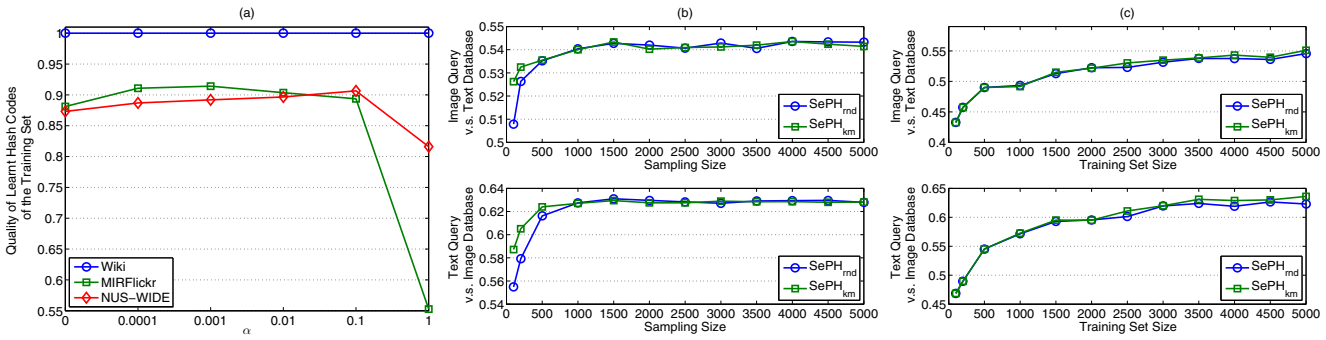

Figure 3. Effects of affecting factors on the proposed SePH. Sub-figure (a) illustrates the effects of the only model parameter α on the quality of the learnt hash codes for the training sets of all datasets. Sub-figure (b) and (c) respectively show the effects of the sampling size for learning kernel logistic regression (*i.e.* hash functions) and those of training set size on the performance of SePH in the largest NUS-WIDE, with the hash code length fixed as 16 bits.

NUS-WIDE firstly increases and then decreases, while that w.r.t. Wiki keeps unchanged with $mAP = 1$, *i.e.* the optimal mAP value. The experimental result is reasonable, because an appropriate non-zero α can help to reduce the quantization loss and make the learnt real-valued hash code matrix (*i.e.* $\hat{H}$) near to the optimal binary one (*i.e.* H), while a large α can lead the optimization process to focus less on minimizing the KL-divergence and thus learn poorer hash codes that cannot well preserve semantic affinities. Additionally, though the empirical value 0.01 is not the optimal setting for α on each dataset, it is still near to the optimal ones and yields better performance than $\alpha = 0$.

4.4. Further Analyses

To reduce the training and predicting costs of kernel logistic regression, in this paper we propose to perform sampling for building smaller kernel feature matrices for spanning the to-be-learnt weighting vectors. In previous experiments, the sampling size is set as 500 on all datasets. Here we conduct experiments on the largest NUS-WIDE to see its effects on the performance of SePH. Similarly, the hash code length is fixed as 16 bits. Then with learnt hash codes on the training set, we vary the sampling size from 100 to $5,000$ and perform random sampling and k-means to learn the corresponding hash functions. For each sampling size, the corresponding cross-view retrieval performance on the query set is measured and compared, as illustrated in Fig. 3(b). It can be seen that with the sampling size increasing, the performance of SePH increases and then converges quickly. Specially, for NUS-WIDE, the performance of SePH begins to converge at a sampling size around 1,000. And even for the empirical setting in our experiments, *i.e.* 500, it can achieve more than 98% of the performance w.r.t. the largest sampling size (*i.e.* 5,000) at around 10% of its costs, as theoretically analysed before. Therefore, performing sampling for learning kernel logistic regression is reasonable. Additionally, we can observe that the superiority of k-means over random sampling is more significant at small sampling sizes (*e.g.* 100). Because in those cases, sampled kernel feature vectors are far from redundant and k-means can probably select better ones.

Moreover, we analyse the effects of the training set size on the performance of SePH. Similarly, with the hash code length fixed as 16 bits, we vary the training set size from 100 to $5,000$, and measure the corresponding performance of SePH on the largest NUS-WIDE, as shown in Fig. 3(c). It can be seen that the performance of SePH keeps increasing and finally tends to converge as the training set size increases. For NUS-WIDE, the performance of SePH begins to converge at a training set size around $3,000$, less than 2% of the retrieval set. It means that SePH can well exploit the limited supervised information of a dataset and can be suitable for large-scale databases, since only the supervised information of a small fraction is needed for training.

Similar analysis results are observed on other datasets.

5. Conclusions

In this paper, we propose a supervised hashing method termed SePH for cross-view retrieval. For training, SePH firstly transforms the given semantic affinities of training data into a probability distribution and approximates it with to-be-learnt hash codes in Hamming space via minimizing the KL-divergence. Then in each view, SePH utilizes kernel logistic regression with a sampling strategy to learn the non-linear projections from features to hash codes. As for out-of-sample extension, the unified hash code of any unseen instance is determined using a novel probabilistic approach, with its predicted hash codes as well as the corresponding output probabilities from different observed views. SePH is validated on three benchmark datasets, and it yields state-of-the-art performance.

6. Acknowledgments

This research was supported by the National Basic Research Project of China (Grant No. 2011CB70700) and the National Natural Science Foundation of China (Grant No. 61271394). The authors would like to thank the anonymous reviewers for their valuable comments.

References

[1] M. Bronstein, A. Bronstein, F. Michel, and N. Paragios. Data fusion through cross-modality metric learning using similarity-sensitive hashing. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[2] T. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng. Nus-wide: A real-world web image database from national university of singapore. In *Proceedings of the ACM International Conference on Image and Video Retrieval*, 2009.

[3] G. Ding, Y. Guo, and J. Zhou. Collective matrix factorization hashing for multimodal data. In *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[4] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[5] D. Hardoon, S. Szedmak, and J. Shawe-taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural Computation*, 16(12), 2004.

[6] K. He, F. Wen, and J. Sun. K-means hashing: An affinity-preserving quantization method for learning binary compact codes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[7] M. Huiskes and M. Lew. The mir flickr retrieval evaluation. In *Proceedings of the 2008 ACM International Conference on Multimedia Information Retrieval*, 2008.

[8] G. Irie, Z. Li, X. Wu, and S. Chang. Locally linear hashing for extracting non-linear manifolds. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[9] S. Kumar and R. Udupa. Learning hash functions for cross-view similarity search. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[10] G. Lin, C. Shen, Q. Shi, A. Hengel, and D. Suter. Fast supervised hashing with decision trees for high-dimensional data. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[11] W. Liu, J. Wang, R. Ji, Y. Jiang, and S. Chang. Supervised hashing with kernels. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[12] J. Pereira, E. Coviello, G. Doyle, N. Rasiwasia, G. Lanckriet, R. Levy, and N. Vasconcelos. On the role of correlation and abstraction in cross-modal multimedia retrieval. *Transactions of Pattern Analysis and Machine Intelligence*, 36(3):521–535, 2014.

[13] F. Shen, C. Shen, Q. Shi, A. Hengel, and Z. Tang. Inductive hashing on manifolds. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[14] J. Song, Y. Yang, Y. Yang, Z. Huang, and H. Shen. Inter-media hashing for large-scale retrieval from heterogeneous data sources. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, 2013.

[15] L. V. and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

[16] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Advances in Neural Information Processing Systems*, 2009.

[17] Z. Yu, F. Wu, Y. Yang, Q. Tian, J. Luo, and Y. Zhuang. Discriminative coupled dictionary hashing for fast cross-media retrieval. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2014.

[18] D. Zhai, H. Chang, Y. Zhen, X. Liu, X. Chen, and W. Gao. Parametric local multimodal hashing for cross-view similarity search. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

[19] D. Zhang and W. Li. Large-scale supervised multimodal hashing with semantic correlation maximization. In *AAAI Conference on Artificial Intelligence*, 2014.

[20] D. Zhang, F. Wang, and L. Si. Composite hashing with multiple information sources. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2011.

[21] D. Zhang, J. Wang, D. Cai, and J. Lu. Self-taught hashing for fast similarity search. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2010.

[22] P. Zhang, W. Zhang, W. Li, and M. Guo. Supervised hashing with latent factor models. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2014.

[23] Y. Zhen and D. Yeung. Co-regularized hashing for multimodal data. In *Advances in Neural Information Processing Systems*, 2012.

[24] Y. Zhen and D. Yeung. A probabilistic model for multimodal hash function learning. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2012.

[25] J. Zhou, G. Ding, and Y. Guo. Latent semantic sparse hashing for cross-modal similarity search. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2014.

[26] J. Zhou, G. Ding, Y. Guo, Q. Liu, and X. Dong. Kernel-based supervised hashing for cross-view similarity search. In *IEEE International Conference on Multimedia and Expo (ICME)*, 2014.

[27] X. Zhu, Z. Huang, H. Shen, and X. Zhao. Linear cross-modal hashing for efficient multimedia search. In *Proceedings of the 21st ACM International Conference on Multimedia*, 2013.