This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2016.2610969, IEEE Transactions on Pattern Analysis and Machine Intelligence

Linear Subspace Ranking Hashing for Cross-modal Retrieval

Kai Li, Guo-jun Qi, Jun Ye, and Kien A. Hua, *Fellow, IEEE*

Abstract—Hashing has attracted a great deal of research in recent years due to its effectiveness for the retrieval and indexing of large-scale high-dimensional multimedia data. In this paper, we propose a novel ranking-based hashing framework that maps data from different modalities into a common Hamming space where the cross-modal similarity can be measured using Hamming distance. Unlike existing cross-modal hashing algorithms where the learned hash functions are binary space partitioning functions, such as the sign and threshold function, the proposed hashing scheme takes advantage of a new class of hash functions closely related to rank correlation measures which are known to be scale-invariant, numerically stable, and highly nonlinear. Specifically, we jointly learn two groups of linear subspaces, one for each modality, so that features' ranking orders in different linear subspaces maximally preserve the cross-modal similarities. We show that the ranking-based hash function has a natural probabilistic approximation which transforms the original highly discontinuous optimization problem into one that can be efficiently solved using simple gradient descent algorithms. The proposed hashing framework is also flexible in the sense that the optimization procedures are not tied up to any specific form of loss function, which is typical for existing cross-modal hashing methods, but rather we can flexibly accommodate different loss functions with minimal changes to the learning steps. We demonstrate through extensive experiments on four widely-used real-world multimodal datasets that the proposed cross-modal hashing method can achieve competitive performance against several state-of-the-arts with only moderate training and testing time.

Index Terms—Cross-modal hashing, large-scale similarity search, image and text retrieval, ranking subspace learning, rank correlation measures, max-order-statistics.

1 Introduction

Thanks to the rapid advancement of information technologies, the last decade has witnessed unprecedented growth in multimedia content generated by all kinds of digital electronic devices, such as digital cameras, mobile phones and tablets etc. With its massive and quickly-increasing volume, multimedia data calls for efficient techniques to support effective indexing and fast similarity search based on semantic content.

Hashing has received considerable attention from researchers in addressing the above problems for its storage and computation efficiency [1]. Typically, hashing algorithms transform high-dimensional data into compact binary codes, resulting in immediate benefits: 1) binary codes take much less storage compared to the original high-dimensional float/double vectors; 2) binary bits can be naturally used for indexing to support fast sublinear search; 3) binary codes enable fast Hamming distance computation based on bit operations which are extremely favored by modern computers.

Most earlier hashing techniques [2, 3, 4, 5, 6, 7] are designed for single-modal data; that is, data can only be queried by an example from the same modality. However, multimedia data usually come in different modalities. For example, an image may be associated with a textual caption such that both of them describe the same semantic object. In this case, it's desirable to use textual queries to search for relevent images and vice versa. Such scenarios entail designing hashing techniques to enable similarity search using data from another modality, which is crucial to many practical applications [8, 9, 10, 11, 12, 13].

Cross-modal hashing is a challenging problem because data from different modalities typically have distinct representations with incomparable space structures and dimensionalities. Existing algorithms [8, 11, 12, 14, 15, 16, 17, 18, 19, 20, 21] generally follow two steps: first, features from different modalities are mapped into a common feature space to minimize some cross-correlation error; second, hash codes are generated by binary partitioning of the feature space obtained through linear or nonlinear transformation of the original features. Different hashing techniques usually differ in the first step, where different error functions are defined. As for the second step, they can be similarly represented as the binary embedding: $h(\mathbf{x}; \mathbf{w}) = sign(F_{\mathbf{w}}(\mathbf{x}))$, where $\mathbf{x}$ is the input vector, $\mathbf{w}$ is the solution to the optimization problem in the first step and $F_{\mathbf{w}}(\cdot)$ is a feature transformation function parameterized by $\mathbf{w}$.

In comparison, we consider a completely different family of hash functions based solely on the ranking ordering of feature dimensions. Such ranking-based hash functions are closely related with rank correlation measures [22], which have been well-deemed as robust measures in many performance evaluation schemes. Ranking-based *randomized* hashing schemes have been exploited in single-modality settings, and representative works in this category include the Winner-Take-all (WTA) Hash [23] and the Min-wise Hash (MinHash) [24]. WTA generates a compact representation of the input features by ranking the random permutations of input features and outputs the index of the maximum feature dimension as the hash code. MinHash is a special case of WTA for binary input features. These ranking-

• *Authors are with the Department of Computer Science, University of Central Florida, Orlando, FL, 32816.*
Corresponding author: Kai Li (kaili@cs.ucf.edu)

based hashing schemes rely on the relative ordering of feature dimensions and are very robust against prevalent noises and variations that do not affect the implicit ordering [23, 24]. However, since those hashing algorithms are data-agnostic and totally based on randomization, it's typically necessary to generate sufficiently long codes through a large number of permutations to achieve desirable performance [23]. Moreover, since the same sequence of permutations is needed to generate comparable hash codes, these methods are only applicable to single-modal data and do not naturally fit into the cross-modal retrieval tasks.

To address the above challenges, we propose to generate hash codes by ranking optimized linear subspaces instead of the random permutations of the input features. Specifically, we learn two groups of linear subspaces jointly, one for each modality, such that the ranking ordering in one subspace is maximally aligned with that of the other. Ranking hash codes learned in this way are highly optimized for cross-modal similarity search and competitive performance can be obtained even with very compact codes. Ranking-based hash functions can be hard to learn in their original form, due to the highly non-convex and discontinuous nature. We present a probabilistic relaxation of the original problem, reformulating it as one that can be flexibly combined with different loss functions and efficiently solved using stochastic gradient descent algorithms, thus making the training process scalable with large-scale datasets.

To sum up, we highlight the major contribution of this paper as follows:

- The proposed method is the first supervised hashing algorithm that exploits ranking-based hash functions in cross-modal similarity search. This method generates optimal linear subspaces where the ranking ordering of different feature dimensions maximally preserve the cross-modal similarities.
- We propose an effective relaxation of the original non-convex and discontinuous problem and reformulate it in a way that can be efficiently solved by stochastic gradient descent algorithms that are known to be scalable with large datasets.
- The proposed hashing scheme can easily accommodate different loss functions with the same learning procedure, leading to a flexible cross-modal hash learning framework.
- We demonstrate through extensive experiments on real-world datasets that the proposed algorithm outperforms state-of-the-art hashing schemes in a number of cross-modal retrieval tasks.

Preliminary results of this work have been published in [25]. We have made the source code available at https://bitbucket.org/kailee880209/cmrsh.

2 Related Work

Hashing for uni-modal data has been extensively studied in the past decade [2, 3, 23, 26, 27]. In contrast, cross-modal hashing starts to receive increasing attention only very recently and is the focus of this review.

Cross-Modal Similarity Sensitive Hashing (CMSSH) [8] and Cross-View Hashing (CVH) [28] are arguably the earliest works on this topic. CMSSH sequentially constructs two groups of linear hash functions and explicitly minimizes the distances between the Hamming space embeddings of data from different modalities. CVH extends the unimodal hashing method, Spectral Hashing (SH) [29], to consider both intra-view and inter-view similarities with a generalized eigenvalue formulation.

Several new methods were proposed soon after CMSSH and CVH. Iterative Multi-View Hashing (IMVH) [30] learns discriminative hash functions by solving a series of binary label assignment problems. Co-Regularized Hashing (CRH) [9] learns single-bit cross-modal hash functions by solving DC (i.e. difference of convex function) programs; and multiple bits are sequentially learned using boosting. The same authors of CRH also propose Multimodal Latent Binary Embedding (MLBE) [31], which takes a probabilistic generative approach and achieves competitive performance. However, the prohibitive computational costs for out-of-sample extensions limit the applications of MLBE to large-scale datasets.

In order to balance performance and computational costs, several new methods are proposed. For example, (PLMH) [32] extends MLBE to learn parameterized hash functions as the linear combination of a small set of anchor points. Similar ideas have also been exploited in Linear Cross-Modal Hashing (LCMH) [33], where a small set of cluster centroids are used in a similar fashion to the anchor points in PLMH. Semantic Correlation Maximization (SCM) [34] integrates semantic label information into a learning procedure with closed-form solutions and scales to large datasets by avoiding the explicit computation of pairwise similarity matrix. Inter-Media Hashing (IMH) [10] incorporates both labeled and unlabeled data to explore correlations among multiple media types from large-scale data sources.

More recently, Sparse Multi-Modal Hashing (SM^2H) [35] is proposed to obtain sparse codesets for data objects across different modalities through joint multi-modal dictionary learning. Latent Semantic Sparse Hashing (LSSH) [11] and Collective Matrix Factorization Hashing (CMFH) [12] use sparse coding and matrix factorization to capture the latent semantic features of different modalities. Supervised Matrix Factorization Hashing (SMFH) [36] also uses matrix factorization, however with the addition of graph-regularization. Semantics-Preserving Hashing (SePH) [18] transforms the similarity matrix into a joint probability distribution and approximates the distribution using nonlinear functions of the pairwise Hamming distance. Quantized Correlation Hashing (QCH) [20] considers both intra-modality quantization loss and inter-modality correlation in a single multi-modal objective function. The multi-modal objective function is further transformed to a unimodal formulation and optimized through an alternative procedure. Similar quantization-based cross-modal hashing algorithms include Cross-Modal Collaborative Quantization (CMCQ) [37] and Alternating Co-Quantization (ACQ) [38]. Semantic Topic Multimodal Hashing (STMH) [21] learns a common feature subspace from multimodal semantic concepts, and encode a hash bit by examining the existence of a concept.

Neural networks and deep learning have also been used in cross-modal hashing [14, 16, 39, 40, 41]. Specifically, end-to-end deep cross-modal hashing frameworks, such as Deep Cross-modal Hashing (DCMH) [42] are starting to receive

attention lately. However, in contrast to the conventional hash learning research, where the focus is to learn good hash functions given certain feature representations, end-to-end hashing focuses on the seamless combination of feature learning and hash learning, in a way that the feature representations can be optimized for hash learning through error back-propagation. The competitive performance shown in [42] demonstrates the efficacy of combining hash learning and feature learning methods.

To sum up, most of the existing hash learning methods are restricted to a specific class of hash functions featuring binary partitioning of the projected feature spaces. To the best of our knowledge, there has been no previous work exploring ranking-based hash functions in cross-modal hash learning.

3 Linear Subspace Ranking Hashing

3.1 Mathematical Notations

Suppose we have the data sets from two modalities $\mathcal{X}$ and $\mathcal{Y}$. Let $\mathcal{D}_\mathcal{X}$ be a set of $d_\mathcal{X}$-dimensional data points $\{\mathbf{x}_i\}_{i=1}^{N_\mathcal{X}}$ from modality $\mathcal{X}$ and $\mathcal{D}_\mathcal{Y}$ be a set of $d_\mathcal{Y}$-dimensional data points $\{\mathbf{y}_i\}_{i=1}^{N_\mathcal{Y}}$ from modality $\mathcal{Y}$. In addition, we have a set of inter-modality similarity labels $\mathbf{S} = \{s_{ij}\} \in \{1, 0\}^{N_\mathcal{X} \times N_\mathcal{Y}}$ indicating whether the cross-modal pair $(\mathbf{x}_i, \mathbf{y}_j)$ describe the same concept or not. The similarity labels can be obtained by comparing data points' semantic labels; or by thresholding certain metric distances if annotations are not available. Our objective is to learn two sets of hash functions $H_* = \{h_*^{(l)}\}_{l=1}^{L}$ with '$*$' being a place holder for $\mathcal{X}$ or $\mathcal{Y}$, so that data from both modalities can be projected into a common Hamming space.

3.2 Ranking-based Hash Function

Most hashing methods choose binary space partitioning functions (e.g sign() or threshold()) as their hash function. In contrast, we consider a family of hash functions based on the max-order-statistics of feature projections onto a K-dimensional linear subspace. Formally, the hash function $h_*(\cdot)$ is defined as

$$h_*(\mathbf{z}_*; \mathbf{W}_*) = \arg \max_{1 \le k \le K} \mathbf{w}_{*_k}^T \mathbf{z}_*, \tag{1}$$

where $\mathbf{z}_* \in \mathcal{D}_*$ and $\mathbf{W}_* = [\mathbf{w}_{*_1}\ \mathbf{w}_{*_2}\ \cdots \mathbf{w}_{*_K}]^T \in \mathbb{R}^{K \times d_*}$ defines a K-dimensional linear subspace for ranking features. Note that the above hash function generates a single hash code and we have omitted the superscript 'l' to avoid notational clutter.

Briefly, the hash function defined in (1) encodes an input data point as the index of the maximum feature dimension in a K-dimensional linear subspace with $\mathbf{W}_*$ as the basis. This encoding is entirely based on the relative ordering of feature dimensions rather than the precise numerical values. It acts as a nonlinear feature space transformation and generates an ordinal space embedding. Such ordinal embeddings share a similar spirit with rank correlation measures [22], which evaluate the tendency for two ranking orders to match each other. Such measures are robust against noise and variation since the metric values do not change as long as the implicit ordering of two rankings remain unchanged. Compared to known rank correlation methods, the max-order-statistics in (1) provide another level of stability since it's based on a partial ordering of the ranked feature dimensions. To see this, the hash codes generated by (1) only depend on pairwise orders between the maximum element and the remaining feature dimensions.

Obviously, each hash code generated in this way requires only $\lceil \log_2 K \rceil$ binary bits of storage and therefore a length-L K-ary code word can be compactly represented using $L \times \lceil \log_2 K \rceil$ binary bits. Different values of K lead to different emphasis on global or local comparisons among feature dimensions. For instance, $K = 2$ leads to pairwise orders among the projected feature dimensions, while a larger K results in higher-order comparison among the features. In this spirit, larger values of K place emphasis on a more global comparison among the features in the subspace.

3.3 The Connection with WTA

The proposed ranking-based multimodal hashing method is closely related to WTA Hash [23]. Indeed, WTA Hash is a special case of the hash function defined in (1) when the linear subspaces are defined by axis-aligned projections. To see this, note that each bit of WTA code is specified by a window size K and a permutation π. For an input data point, the permutation π reorders its feature dimensions and outputs the index of the maximum value in the first K dimensions. This is equivalent to setting the $\mathbf{w}_{*k}$s in (1) to columns randomly chosen from a $d_* \times d_*$ identity matrix. The restriction to ranking only original feature dimensions greatly limits the flexibility of WTA to discover the potential discriminativity of ranking properties hidden in arbitrary linear subspaces. In addition, since WTA is based on random selection of feature dimensions, the hash codes obtained in heterogeneous feature spaces are incomparable, thus making WTA inapplicable to multi-modal hashing. In comparison, the generalized ranking-based hash function can be flexibly tuned to rank arbitrary feature subspaces and discover the rank correlation measures across heterogeneous data modalities.

3.4 Problem Definition

For each cross-modal training pair $(\mathbf{x}_i, \mathbf{y}_j)$ with a similarity label s_{ij}, we define an empirical loss term incurred by the hash function in (1) as

$$\ell(h_\mathcal{X}^i, h_\mathcal{Y}^j, s_{ij}) = \begin{cases} \mathrm{I}(h_\mathcal{X}^i \neq h_\mathcal{Y}^j), & s_{ij} = 1 \\ \lambda \mathrm{I}(h_\mathcal{X}^i = h_\mathcal{Y}^j), & s_{ij} = 0 \end{cases} \tag{2}$$

where $\mathrm{I}(\cdot)$ is the indicator function that equals 1 when the condition holds and 0 otherwise; $h_\mathcal{X}^i$ and $h_\mathcal{Y}^j$ are short for $h_\mathcal{X}(\mathbf{x}_i; \mathbf{W}_\mathcal{X})$ and $h_\mathcal{Y}(\mathbf{y}_j; \mathbf{W}_\mathcal{Y})$; and λ is a hyper-parameter controlling the relative penalty of false positive pairs.

Intuitively, this error function penalizes similar pairs with different hash codes or dissimilar pairs with the same hash code. The overall learning objective is to find $\mathbf{W}_\mathcal{X}$ and $\mathbf{W}_\mathcal{Y}$ that minimize the aggregate loss over all the training pairs,

$$\mathcal{L}(\mathbf{W}_\mathcal{X}, \mathbf{W}_\mathcal{Y}) = \sum_{s_{ij} \in \mathcal{S}} \ell(h_\mathcal{X}^i, h_\mathcal{Y}^j, s_{ij}). \tag{3}$$

Note that $\mathbf{W}_\mathcal{X}$ and $\mathbf{W}_\mathcal{Y}$ factor into the above objective function because $h_\mathcal{X}^i$ and $h_\mathcal{Y}^j$ are functions of $\mathbf{W}_\mathcal{X}$ and $\mathbf{W}_\mathcal{Y}$.

3.5 Reformulation

The objective function in (3) is hard to optimize due to the $\arg\max$ terms, which are typically non-convex and highly discontinuous. We seek to reformulate the problem first and then approximate it with a continuous formulation that can be solved much more easily.

Note that the ranking-based hash function defined in (1) can be equivalently formulated as

$$\begin{aligned} &\mathbf{h}(\mathbf{z};\mathbf{W}) = \arg\max_{\mathbf{g}} \mathbf{g}^T\mathbf{W}\mathbf{z}, \\ &\text{s.t. } \mathbf{g} \in \{0,1\}^K, \mathbf{1}^T\mathbf{g} = 1, \end{aligned} \tag{4}$$

where we have dropped the subscript placeholder for $\mathcal{X}$ and $\mathcal{Y}$ for notation simplicity. The 1-of-K coding scheme is used to represent the K-ary hash code for an input feature. To see the equivalence, the constrained binary code output by (4) acts as a dimension selector of the maximum dimension of $\mathbf{W}\mathbf{z}$, where the maximum value can only be obtained by setting the bit of $\mathbf{g}$ corresponding to the maximum dimension to 1. We remark that the reformulation is only for formulation convenience. It does not increase the storage cost of a ranking hash code, i.e., only $\lceil \log_2 K \rceil$ bits are needed to store a K-ary code word.

Actually, the vectorizd representation in (4) can be approximated using the softmax function

$$\mathbf{h}(\mathbf{z};\mathbf{W}) \approx \sigma(\mathbf{W}\mathbf{z}), \tag{5}$$

where the function $\delta(\mathbf{x})$ takes a K-dimensional vector $\mathbf{x}$ as input and outputs a vector of the same dimensionality. Specifically, the j^{th} dimension of the output vector is defined as

$$\sigma(\mathbf{x})_j = \frac{\exp \alpha x_j}{\sum_{k=1}^{K} \exp \alpha x_k}, \text{ for } j = 1, \cdots, K, \tag{6}$$

where α controls the smoothness of the approximation and x_j denotes the j^{th} entry of $\mathbf{x}$.

The above approximation has a natural probabilistic interpretation. Each entry in the output vector represents the probability of that dimension being the maximum. When $\alpha \to \infty$, the softmax vector converges to the binary indicator vector.

The probabilistic approximation of the hash function also lends itself to a continuous relaxation of the error function in (2). Formally, consider $\mathbf{W}_\mathcal{X}$ and $\mathbf{W}_\mathcal{Y}$ are given. Let $(h^i_\mathcal{X}, h^j_\mathcal{Y})$ be the K-ary ranking hash codes of a cross-modal pair $(\mathbf{x}_i, \mathbf{y}_j)$, and let $(\mathbf{p}_i, \mathbf{q}_j)$ be the softmax vectors (i.e. $\mathbf{p}_i \equiv \sigma(\mathbf{W}_\mathcal{X}\mathbf{x}_i)$ and $\mathbf{q}_j \equiv \sigma(\mathbf{W}_\mathcal{Y}\mathbf{y}_j)$). Following the probabilistic interpretation of the softmax vector, $h^i_\mathcal{X}$ and $h^i_\mathcal{Y}$ can be regarded as two *independent* discrete random variables with the probability distribution

$$\begin{aligned} P(h^i_\mathcal{X} = k|\mathbf{W}_\mathcal{X}) &= p_{ik} \\ P(h^j_\mathcal{Y} = k|\mathbf{W}_\mathcal{Y}) &= q_{jk}, \end{aligned} \tag{7}$$

where $k \in \{0, \cdots, K-1\}$, and p_{ik} and q_{jk} are the k^{th} dimension of $\mathbf{p}_i$ and $\mathbf{q}_j$ respectively. The probability that two hash codes take the same value, denoted as π_{ij}, can be computed as

$$\begin{aligned} \pi_{ij} \equiv& P(h^i_\mathcal{X} = h^j_\mathcal{Y}|\mathbf{W}_\mathcal{X}, \mathbf{W}_\mathcal{Y}) \\ =& \sum_{k=1}^{K} P(h^i_\mathcal{X} = k|\mathbf{W}_\mathcal{X}) P(h^j_\mathcal{Y} = k|\mathbf{W}_\mathcal{Y}) \\ =& \sum_{k=1}^{K} p_{ik} q_{jk} = \mathbf{p}_i^T \mathbf{q}_j \end{aligned} \tag{8}$$

Based on (8) and (2), one can compute the expected loss for a similar cross-modal pair (i.e. $s_{ij} = 1$) as

$$\mathbb{E}_\alpha[\ell_{ij}] = P(h^i_\mathcal{X} \neq h^j_\mathcal{Y}|\mathbf{W}_\mathcal{X}, \mathbf{W}_\mathcal{Y})\, \mathrm{I}(h^i_\mathcal{X} \neq h^j_\mathcal{Y}) = 1 - \pi_{ij},$$

where ℓ_{ij} is short for $\ell(h^i_\mathcal{X}, h^j_\mathcal{Y}, s_{ij})$. A similar formulation can be applied to dissimilar pairs, resulting in the following approximation to the error function in (2):

$$\tilde{\ell}_{ij} = \begin{cases} 1 - \pi_{ij}, & s_{ij} = 1 \\ \lambda \pi_{ij}, & s_{ij} = 0, \end{cases} \tag{9}$$

By using (9), the overall objective function can be reformulated accordingly:

$$\begin{aligned} \tilde{\mathcal{L}_\alpha}(\mathbf{W}_\mathcal{X}, \mathbf{W}_\mathcal{Y}) =& \sum_{s_{ij}=1} (1 - \pi_{ij}) + \sum_{s_{ij}=0} \lambda \pi_{ij} \\ =& \sum_{s_{ij} \in \mathbf{S}} a_{ij} \mathbf{p}_i^T \mathbf{q}_j + const. \\ =& \operatorname{trace}\left(\mathbf{P}\mathbf{A}\mathbf{Q}^T\right) + const, \end{aligned} \tag{10}$$

where $\mathbf{P} = [\mathbf{p}_1 \ \cdots \ \mathbf{p}_{N_\mathcal{X}}]$ and $\mathbf{Q} = [\mathbf{q}_1 \ \cdots \ \mathbf{q}_{N_\mathcal{Y}}]$ are K-by-N matrices with softmax vectors in each column, and the entries of the $N_\mathcal{X}$-by-$N_\mathcal{Y}$ matrix $\mathbf{A}$ are defined as

$$a_{ij} = \lambda - (\lambda + 1) s_{ij}. \tag{11}$$

In sum, we aim to solve the following relaxed problem:

$$\min_{\mathbf{W}_\mathcal{X}, \mathbf{W}_\mathcal{Y}} \tilde{\mathcal{L}_\alpha} = \operatorname{trace}\left(\mathbf{P}\mathbf{A}\mathbf{Q}^T\right) \tag{12}$$

where $\mathbf{P}$, $\mathbf{A}$ and $\mathbf{Q}$ are as defined in (10).

3.6 Optimization

Intuitively, our objective in (3) is to find two subspaces for each modality, such that the partial ranking ordering maximally correlates with the cross-modal similarity labels. It's not hard to see that (12) achieves the same goal by driving $\mathbf{p}_i^T \mathbf{q}_j$ towards 1 for similar pairs and 0 for dissimilar pairs. In addition, one can easily verify the following relationship between (3) and (10):

$$\lim_{\alpha \to \infty} \tilde{\mathcal{L}_\alpha}(\mathbf{W}_\mathcal{X}, \mathbf{W}_\mathcal{Y}) = \mathcal{L}(\mathbf{W}_\mathcal{X}, \mathbf{W}_\mathcal{Y}). \tag{13}$$

Note that the approximation doesn't change the non-convex nature of the problem. However, the new objective function in (12) is continuous and open to well-established gradient descent algorithms. Specifically, the sum of loss form in (10) leads to a straightforward stochastic gradient descent (SGD) algorithm which approximates the expected gradient with a single pair or a mini-batch. Since the loss

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2016.2610969, IEEE Transactions on Pattern Analysis and Machine Intelligence

Algorithm 1: Linear Subspace Ranking Hashing

Input: Data $\mathbf{X}$, $\mathbf{Y}$ and cross-modal similarity labels $\mathbf{S}$.
Output: Linear projections $\mathbf{W}_{\mathcal{X}}$ and $\mathbf{W}_{\mathcal{Y}}$.

Initialization: Set $\mathbf{W}_{\mathcal{X}}$ and $\mathbf{W}_{\mathcal{Y}}$ to random values from Gaussian distribution
repeat
  Randomly select a training batch $\mathbf{X}_b$, $\mathbf{Y}_b$ and obtain the batchwise similarity labels $\mathbf{S}_b$ accordingly
  Set $\mathbf{A} = \lambda\mathbf{E} - (\lambda+1)\mathbf{S}_b$
  `/* E is a matrix of ones. */`
  Compute $\mathbf{P}$ and $\mathbf{Q}$ by applying the softmax function to each column of $\mathbf{W}_{\mathcal{X}}\mathbf{X}_b$ and $\mathbf{W}_{\mathcal{Y}}\mathbf{Y}_b$
  Set $\mathbf{Q}_s = \mathbf{Q}\mathbf{A}^T$, $\mathbf{P}_s = \mathbf{P}\mathbf{A}^T$
  Update projection matrix $\mathbf{W}_{\mathcal{X}}$ and $\mathbf{W}_{\mathcal{Y}}$ according to equation (15)
until *Convergence*

function is the linear combination of π_{ij}'s, we therefore only compute the derivatives of π_{ij} as follows

$$\begin{aligned}\frac{\partial \pi_{ij}}{\partial \mathbf{W}_{\mathcal{X}}} &= [\mathbf{p}_i \circ \mathbf{q}_j - (\mathbf{p}_i^T\mathbf{q}_j)\mathbf{p}_i]\mathbf{x}_i^T \\ \frac{\partial \pi_{ij}}{\partial \mathbf{W}_{\mathcal{Y}}} &= [\mathbf{q}_j \circ \mathbf{p}_i - (\mathbf{q}_j^T\mathbf{p}_i)\mathbf{q}_j]\mathbf{y}_j^T,\end{aligned} \tag{14}$$

where '$\circ$' stands for the element-wise Hadamard product. The above gradients can be used to update the weights when data samples are presented in a streaming fashion. When all training data are available, mini-batchs are normally used since they lead to more stable convergence. In detail, let $\mathbf{X}_b$ and $\mathbf{Y}_b$ be two mini-batches randomly sampled from $D_{\mathcal{X}}$ and $D_{\mathcal{Y}}$ respectively. By summing pairwise gradients over the batches, we obtain the following equations for weights update

$$\begin{aligned}\mathbf{W}_{\mathcal{X}} &\leftarrow \mathbf{W}_{\mathcal{X}} - \eta[\mathbf{P} \circ \mathbf{Q}_s - \mathbf{P}\,\mathrm{diag}(\mathbf{Q}_s^T\mathbf{P})]\mathbf{X}_b^T \\ \mathbf{W}_{\mathcal{Y}} &\leftarrow \mathbf{W}_{\mathcal{Y}} - \eta[\mathbf{Q} \circ \mathbf{P}_s - \mathbf{Q}\,\mathrm{diag}(\mathbf{P}_s^T\mathbf{Q})]\mathbf{Y}_b^T,\end{aligned} \tag{15}$$

where the 'diag' operator outputs a diagonal matrix by retaining the diagonal entries of a square input matrix, $\mathbf{Q}_s = \mathbf{Q}\mathbf{A}^T$, $\mathbf{P}_s = \mathbf{P}\mathbf{A}$ and η is the learning rate. Note that the notations of $\mathbf{P}, \mathbf{A}$ and $\mathbf{Q}$ are the same as in (10) except that they are defined over the mini-batch.

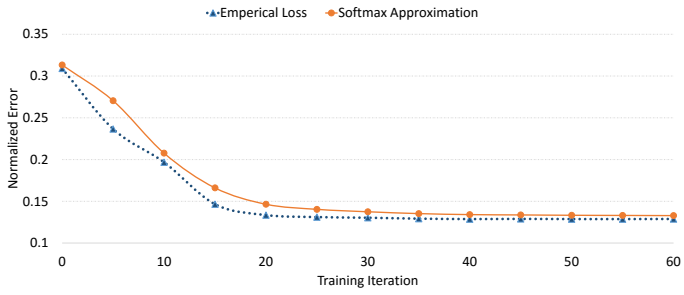


Fig. 1: An example of the empirical loss (3) and its softmax approximation (12) as a function of the training iteration number. The update rules in (15) are used with mini-batch size set to 500. The algorithm converges in less then 50 iterations.

Algorithm 2: Sequential Learning of Multiple Codes

Input: Data $\mathbf{X}$, $\mathbf{Y}$ and cross-modal similarity labels $\mathbf{S}$.
Output: Projections $\{\mathbf{W}_{\mathcal{X}}^{(l)}\}_{l=1}^{L}$ and $\{\mathbf{W}_{\mathcal{Y}}^{(l)}\}_{l=1}^{L}$.

Initialization: Set the weight ω_{ij} of all pairs to one
for $l = 1$ ***to*** L **do**
  Obtain $\mathbf{W}_{\mathcal{X}}^{(l)}$ and $\mathbf{W}_{\mathcal{Y}}^{(l)}$ using Algorithm 1
  Compute hash codes for all samples using (1)
  Set $\epsilon_l = \mathcal{L}(\mathbf{W}_{\mathcal{X}}, \mathbf{W}_{\mathcal{Y}})/(N_{\mathcal{X}} \cdot N_{\mathcal{Y}})$
  Evaluate the quantity $\sigma = \ln(1/\epsilon_l - 1)$
  Update the weighting coefficients using $\omega_{ij}^{(l+1)} = \omega_{ij}^{(l)} \exp[\sigma \cdot \ell(h_{\mathcal{X}}^i, h_{\mathcal{Y}}^j, s_{ij})]$
  Normalize ω_{ij}'s such that $\sum_{i,j} \omega_{ij}^{(l+1)} = N_{\mathcal{X}} \cdot N_{\mathcal{Y}}$
end

3.7 Learning multiple hash codes

The equations in Section 3.6 are used to learn the hash function for one code at a time. The non-convex nature of the objective function means that there are typically multiple local minima instead of a global one. Therefore, a straightforward way to learn L ranking hash codes is to repeat the procedures L times with different random initializations. This naive method takes advantage of multiple local minima to reveal complementary ranking structure. However, independently learned hash functions may be redundant because they may correspond to the same local minima.

In order to minimize code redundancy, we propose to use Adaboost to learn multiple hash functions sequentially. In detail, each training pair is associated with a weight ω_{ij} which is initialized to 1 in the beginning. The weights are updated based on whether its similarity label is correctly predicted using the current hash function. The updated weights are then used to learn the next hash function. Since boosting is not the contribution of this paper, we do not elaborate on it. More details of this standard ensemble learning method can be found in [43].

Interestingly, the introduction of pairwise weights doesn't change the update equations in (15) except that $\mathbf{A}$'s entries are redefined as

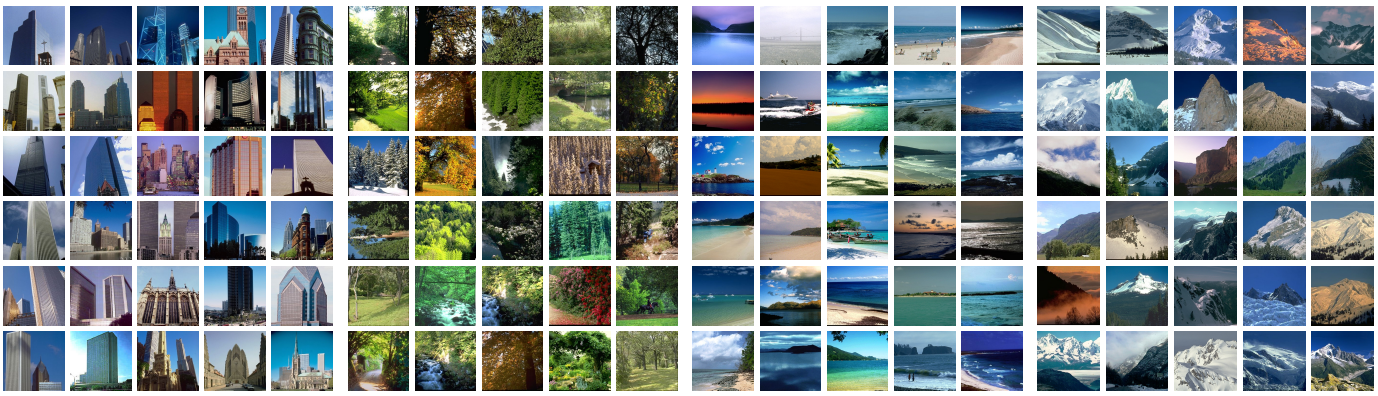
$$a_{ij} = [\lambda - (\lambda+1)s_{ij}]\omega_{ij}. \tag{16}$$

We note that one may learn more than L hash functions and select the best L based on the training error, which may potentially leads to even better results. However, this is not in the scope of this paper and thus will not be discussed in detail here.

3.8 Discussion of different loss functions

Unlike most other cross-modal hashing algorithms, where the loss functions are deeply coupled to the problem formulations and optimization process, our ranking-based hash learning framework can easily accommodate different loss functions. Here we show the flexibility of the proposed method in accommodating different loss functions with minimal changes to the learning procedures. We will also evaluate them in the experimental section.

In the following discussion of different loss functions, we do not consider the trade-off parameter λ or pair's weights



(a) building (b) tree (c) sea (d) mountain

Fig. 2: Example of textual queries on the labelme image database. From (a) to (b), the query keywords are 'building', 'tree', 'sea' and 'mountain' respectively. The code length of LSRH is set to 30 bits and the top 30 results are shown. Images in the grid are ordered from left to right and top to bottom based on the Hamming distance of their hash codes to the hash code of the textual query.

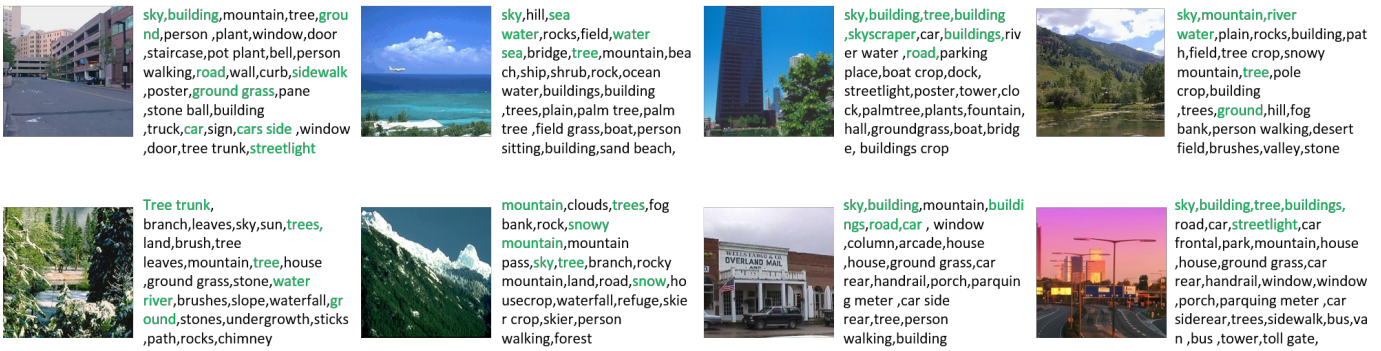


Fig. 3: Example of image annotations by using images to query tags. The code length of LSRH is set to 30 bits and approximately 30 of the most relevant tags are shown. The ground truth for each image is shown in green. The order of the tags is based on the Hamming distance between hash codes.

for notation simplicity; it's straightforward to incorporate them into the formulation.

LSRH-L1 To start with, the loss function in (9) depends only on π_{ij} and s_{ij}, and is equivalent to the L_1 loss.

$$f_{l_1}(\pi_{ij}, s_{ij}) = |\pi_{ij} - s_{ij}|. \tag{17}$$

This is the default loss function used in LSRH. Actually, by representing the loss function in L_1 form, one can easily discover its resemblance to the classic logistic regression problem with π_{ij} as the prediction and s_{ij}s as the target variable. Such observation allows us to easily extend to other loss functions.

LSRH-L2 The L2 loss punishes the prediction's deviation from target by imposing a squared error

$$f_{l_2}(\pi_{ij}, s_{ij}) = (\pi_{ij} - s_{ij})^2. \tag{18}$$

LSRH-Exp The exponential loss function is defined as

$$f_{exp}(\pi_{ij}, s_{ij}) = \exp\{-[2(\pi_{ij} - 0.5)][2(s_{ij} - 0.5)]\}. \tag{19}$$

Here we have applied the mapping from $\pi_{ij} \to 2(\pi_{ij} - 0.5)$ to transform the range of the values from $[0, 1]$ to $[-1, 1]$. Similar operations are applied to s_{ij}.

LSRH-Hinge The hinge loss uses different punishment rules for similar and dissimilar pairs

$$\tilde{\ell}_{ij} = \begin{cases} \max(0.5 - \pi_{ij}, 0), & s_{ij} = 1 \\ \lambda(\pi_{ij} - 0), & s_{ij} = 0. \end{cases} \tag{20}$$

Note that the Hinge loss pushes the similarity scores of similar pairs to be at least 0.5. This is reasonable as similar pairs are only required to have a sufficiently large similarity score, but not necessarily the maximum.

Although those loss functions seem very different in their expressions, all of the optimization steps in Algorithm 1 remain the same except for line 4. Specifically, line 4 of Algorithm 1 can be substituted with the following equations when the corresponding loss functions are used:

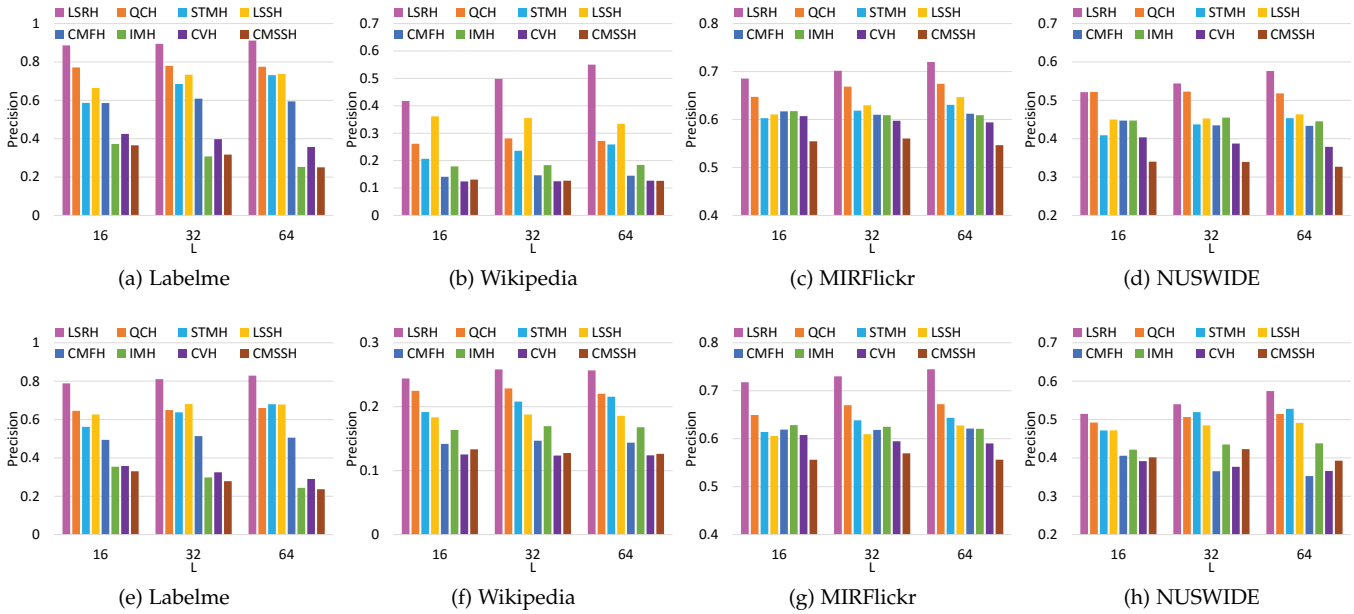
$$\begin{aligned} \mathbf{A}_{l_2} &= 2(\Pi - \mathbf{S}) \\ \mathbf{A}_{exp} &= 2(\mathbf{E} - 2\mathbf{S}) \circ f_{exp}(\Pi, \mathbf{S}) \\ \mathbf{A}_{hinge} &= (\mathbf{E} - \mathbf{S}) + \mathbf{S} \circ [\mathrm{I}(\Pi > 0.5) - \mathbf{E}], \end{aligned} \tag{21}$$

where Π denotes the matrix formed by π_{ij}, $f_{exp}()$ and $\mathrm{I}()$ apply to the input matrix in an element-wise fashion, and $\mathbf{E}$ is a matrix of all ones.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2016.2610969, IEEE Transactions on Pattern Analysis and Machine Intelligence

TABLE 1: Cross-modal mAP results of the proposed LSRH and compared baselines on all of the benchmark datasets. The length of the hash code is varied from 16 bits to 64 bits and the mAP of the top 50 neighbors are reported (i.e. R = 50). The best results are shown in bold. Our LSRH outperforms all baselines in almost all datasets and benchmarks.

	Labelme			Wikipedia			MIRFlickr			NUSWIDE		
Method	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
Text query image												
CVH	0.5630	0.5555	0.5481	0.1931	0.1982	0.2083	0.6434	0.6368	0.6291	0.4466	0.4395	0.4351
CMSSH	0.4369	0.3760	0.3153	0.1802	0.1768	0.1918	0.5997	0.5688	0.5835	0.4080	0.3927	0.3822
IMH	0.4958	0.4202	0.3679	0.2642	0.2703	0.2781	0.6406	0.6416	0.6383	0.4950	0.4975	0.4885
LSSH	0.7408	0.7913	0.8085	0.5002	0.5220	0.5168	0.6430	0.6707	0.6908	0.5013	0.5066	0.5299
CMFH	0.6938	0.7285	0.7330	0.2174	0.2265	0.2290	0.6510	0.6444	0.6461	0.4960	0.4831	0.4824
QCH	0.8151	0.8314	0.8246	0.3420	0.3908	0.3839	0.6602	0.6899	0.6988	**0.5562**	0.5584	0.5565
STMH	0.6487	0.7484	0.7874	0.2924	0.3251	0.3772	0.6315	0.6500	0.6596	0.4459	0.4797	0.4955
LSRH	**0.8883**	**0.8989**	**0.9153**	**0.5459**	**0.6626**	**0.7258**	**0.7108**	**0.7223**	**0.7450**	0.5525	**0.5701**	**0.6068**
Image query text												
CVH	0.4574	0.4191	0.3946	0.1930	0.1865	0.1885	0.6381	0.6301	0.6276	0.4529	0.4356	0.4250
CMSSH	0.3857	0.3229	0.3022	0.1886	0.1749	0.1702	0.5890	0.6069	0.5790	0.4823	0.4833	0.4731
IMH	0.4447	0.3943	0.3401	0.2290	0.2331	0.2275	0.6615	0.6576	0.6554	0.4657	0.4815	0.4903
LSSH	0.6977	0.7317	0.7417	0.2284	0.2355	0.2422	0.6368	0.6421	0.6616	0.5201	0.5318	0.5372
CMFH	0.5835	0.6159	0.6187	0.2045	0.2161	0.2148	0.6528	0.6542	0.6590	0.4648	0.4249	0.4087
QCH	0.6727	0.6788	0.6899	0.2582	0.2568	0.2510	0.6595	0.7048	0.7033	0.5295	0.5463	0.5531
STMH	0.6098	0.6885	0.7310	0.2366	0.2505	0.2616	0.6387	0.6684	0.6750	0.5165	0.5617	0.5696
LSRH	**0.8048**	**0.8211**	**0.8376**	**0.2707**	**0.2816**	**0.2914**	**0.7395**	**0.7529**	**0.7682**	**0.5450**	**0.5724**	**0.6012**



(a) Labelme (b) Wikipedia (c) MIRFlickr (d) NUSWIDE

(e) Labelme (f) Wikipedia (g) MIRFlickr (h) NUSWIDE

Fig. 4: Top-100 precision of different methods on all datasets, with the hash code varying from 16 bits to 64 bits. The first row shows the results of text-query-image and the second row shows that of image-query-text.

4 Experiments

4.1 Datasets

To evaluate the proposed algorithm, we choose four widely-used multimodal datasets: Wikipedia [12, 34], Labelme [11], MIRFLICKR [12, 18] and NUS-WIDE [11, 12, 20, 34]. The statistics of those datasets are shown in Table 2 and following are brief descriptions of each dataset.

Wiki. The wiki [44] dataset, crawled from Wikipedia's "featured articles", consists of 2,866 documents which are image-text pairs and annotated with semantic labels of 10 categories. Each image in this dataset is represented as a 128-D bag-of-SIFT feature vector. For text documents, we extract the 1000-D tf-idf features over the most representative words.

LabelMe.[1] The LabelMe dataset [45] consists of 2688 images annotated by the objects' textual tags contained in them, such as "forest" and "mountain". Tags that occurs less than 3 times are discarded, resulting in 245 unique remaining tags. Each image is labeled as one of eight unique outdoor scenes, such as coast, forest and highway. Each image in this dataset is represented by a 512-D GIST vector and the corresponding textual tags are represented by the index vectors of selected tags.

1. http://people.csail.mit.edu/torralba/code/spatialenvelope/

MIRFLICKR.[2] The MIRFLICKR dataset [46] contains 25,000 images with associated textual tags. Each image-text pair is associated with one or more of 24 semantic labels. Tags that appear less than 20 times are first removed and then instances without tags or annotations are removed, resulting in 16738 instances remaining. Images in this dataset are described by 150-D edge histograms and the texts are represented as 500-D feature vectors obtained by applying PCA to the binary tagging vector. Instances are considered as similar if they share at least one common label.

NUS-WIDE.[3] The NUS-WIDE dataset [47] is a real-world image dataset with $269,648$ images. Each image has a number of textual tags and is labeled with one or more image concepts out of 81 concepts. We select the $186,577$ image-tag pairs belonging to the 10 largest concepts. In this dataset, the images are represented by 500-D bag-of-visual-words (BOVW) and the image tags are represented by 1000-D tag occurrence feature vectors.

TABLE 2: Statistics of benchmark datasets

Dataset	Features		Classes	Size	Queries
	Image	Text			
Labelme	512	245	8	2688	573
Wikipedia	128	1000	10	2866	693
MIRFlickr	150	500	24	16738	836
NUSWIDE	500	1000	10	186577	2000

4.2 Baselines

We have compared the proposed LSRH with seven well-known cross-modal hashing methods: Cross-view Hashing (CVH) [28], Cross-modal Similarity Sensitive Hashing (CMSSH) [8], Inter-media Hashing (IMH) [10], Latent Semantic Sparse Hashing (LSSH) [11], Collective Matrix Factorization Hashing (CMFH) [12], Semantic Topic Multimodal Hashing (STMH) [21] and Quantization Correlation Hashing (QCH) [20]. Those algorithms have been briefly introduced in Section 2 and are considered to be the current state-of-the-arts in cross-modal hash learning. The parameters for all the compared algorithms are either default ones or chosen according to the suggestions of the original papers to give the best performance.

4.3 Evaluation metrics

We evaluate the retrieval performance of both text-query-image and image-query-text. Specifically, we follow the widely used metrics [9, 10, 11, 20, 30]: mean Average Precision (mAP), top-k precision and precision-recall for both retrieval tasks. Those evaluation metrics are defined as follows:

Top-k precision The top-k precision is defined as the ratio of relevant items among the retrieved top k instances in terms of Hamming distance. This metric is averaged over all queries in our evaluation.

mAP The mean average precision is defined as

$$\mathrm{mAP} = \frac{1}{Q}\sum_{q=1}^{Q}\frac{1}{R}\sum_{r=1}^{R}P_q(r)\delta_q(r), \tag{22}$$

where Q is the size of the query set, $P_q(r)$ denotes the top-k precision of the q^{th} query, and $\delta_q(r)$ indicates whether the k^{th} data item is relevant to the q^{th} query.

Precision-recall Precision-recall reflects the precision values at different recall levels and it's a good indicator of the overall performance of different algorithms. Typically, the area under the precision-recall curve is computed and a larger value indicates better performance.

4.4 Experiment settings

We follow previous work [11, 12, 20, 21] in choosing the training set and query set. In detail, for Labelme and Wikipedia, 20% of the data points are randomly selected as the query set, and the remaining data are used as the training set and retrieval database. For MIRFlickr and NUSWIDE, we randomly select approximately 5% and 1% of the dataset as queries respectively. The remaining data are used as the database for cross-modal retrieval. Moreover, we randomly select 5000 image-text pairs from the database for hash learning and apply the learned hash functions to the entire database to generate the hash codes. Such practice has been widely used in hash learning research [8, 11, 12, 18, 20, 21, 28] because it simulates real-world scenarios where the labeled data are limited compared to the entire data corpus, and this can well-demonstrate the out-of-sample extension capability of different hashing methods.

LSRH takes a single primary parameter: the subspace dimension K; and two hyper-parameters λ and β. We choose those parameters by using 5-fold cross-validation on a held-out subset in the training set. Specifically, we use linear search in log scale for K, and fix it to 4 for all the experiments. The effect of subspace dimension K will be discussed in detail in Section 4.8. As for λ and β, we use linear search over $\{0.5, 1.0, 2.0\}$ and $\{0.5, 0.8, 1.0\}$ respectively.

In contrast to the binary hashing algorithms, our hashing code is K-ary. Therefore, we set $L = \lfloor N_b/\lceil \log_2 K \rceil \rfloor$ when comparing with other binary hashing codes at N_b bits to ensure fairness. All the experimental results are averaged over 5 independent runs.

4.5 Comparison with baselines

We vary the hash code from 16 bits to 64 bits and record the mAPs of LSRH and baselines on all the benchmark datasets in Table 1. We can observe that LSRH is competitive to or outperforms all the compared methods across different datasets and code lengths. In fact, the average performance advantage of LSRH is more than 5% across the four datasets. Furthermore, when it comes to individual benchmarks, LSRH can beat the best baseline by up to 40%, for example, in the 64-bit text-query-image task on Wikipedia, while most of the baselines are only competitive in some benchmarks or datasets. For instance, STMH is very competitive in the image-query-text task on NUSWIDE, but not as competitive in the text-query-image task on the same dataset; LSSH performs very well in the text-query-image task on Wikipedia, but is not equally good in the the image-query-text task compared to some other baselines.

We find that the performance difference between text-query-image and image-query-text tasks are very close in

2. http://press.liacs.nl/mirflickr/
3. http://lms.comp.nus.edu.sg/research/NUS-WIDE.htm

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2016.2610969, IEEE Transactions on Pattern Analysis and Machine Intelligence

TABLE 3: Performance results of the proposed LSRH and baselines in terms of top-k precision, precision-recall and training/testing time on all of the benchmark datasets. The code length is set to 32 bits in this experiment and the precision of the top 100 neighbors is reported. The precision-recall values are computed as the area under the precision-recall curve. The best results for top-k precision and precision-recall are shown in bold. Our LSRH consistently outperforms all baselines in different performance metrics with only moderate training and testing time.

		Time (s)		Precision (top 100)			Precesion-Recall		
Method	**#Train**	Train	Test	Text query image	Image query text	Average	Text query image	Image query text	Average
					Labelme				
CVH	2151	1.3	0.005	0.3974	0.3250	0.3612	0.2917	0.2462	0.2690
CMSSH	2151	481.0	0.005	0.3172	0.2785	0.2979	0.2359	0.2311	0.2335
IMH	2151	5.5	0.008	0.3076	0.2984	0.3030	0.2270	0.2218	0.2244
LSSH	2151	263.4	14.9	0.7015	0.6805	0.6910	0.6234	0.5602	0.5918
CMFH	2151	10.8	0.005	0.5810	0.5136	0.5473	0.4821	0.3963	0.4392
QCH	2151	143.2	0.005	0.7796	0.6494	0.7145	0.5803	0.5452	0.5628
STMH	2151	3.7	0.2	0.6614	0.6375	0.6323	0.5727	0.5417	0.5572
LSRH	2151	13.7	0.009	**0.8944**	**0.8107**	**0.8526**	**0.8756**	**0.8485**	**0.8626**
					Wikipedia				
CVH	2173	0.08	0.005	0.1246	0.1237	0.1242	0.1147	0.1148	0.1148
CMSSH	2173	623.5	0.005	0.1268	0.1279	0.1207	0.1208	0.1205	0.1160
IMH	2173	5.7	0.01	0.1834	0.1697	0.1766	0.1430	0.1385	0.1408
LSSH	2173	136.3	8.8	0.3554	0.1878	0.2716	0.2470	0.1443	0.1957
CMFH	2173	9.3	0.006	0.1466	0.1469	0.1468	0.1240	0.1247	0.1244
QCH	2173	107.8	0.005	0.2813	0.2286	0.2550	0.2094	0.1882	0.1988
STMH	2173	6.3	0.5	0.2360	0.2081	0.2221	0.1863	0.1669	0.1766
LSRH	2173	14.2	0.02	**0.4983**	**0.2584**	**0.3784**	**0.3902**	**0.2325**	**0.3114**
					MIRFlickr				
CVH	5000	0.05	0.02	0.5974	0.5947	0.5961	0.5720	0.5716	0.5718
CMSSH	5000	350.1	0.02	0.5604	0.5695	0.5650	0.5580	0.5575	0.5578
IMH	5000	41.2	0.04	0.6090	0.6249	0.6170	0.5808	0.5809	0.5809
LSSH	5000	163.7	36.1	0.6297	0.6093	0.6195	0.5784	0.5759	0.5772
CMFH	5000	12.0	0.02	0.6100	0.6183	0.6142	0.5784	0.5787	0.5786
QCH	5000	228.7	0.02	0.6685	0.6698	0.6692	0.6011	0.6026	0.6019
STMH	5000	7.9	1.6	0.6185	0.6384	0.6285	0.5857	0.5842	0.5850
LSRH	5000	26.6	0.03	**0.7016**	**0.7301**	**0.7159**	**0.6688**	**0.6844**	**0.6766**
					NUSWIDE				
CVH	5000	0.8	0.4	0.3874	0.3768	0.3821	0.3462	0.3421	0.3445
CMSSH	5000	1084.4	0.4	0.3395	0.4229	0.3812	0.3211	0.3252	0.3418
IMH	5000	42.1	0.7	0.4549	0.4349	0.4449	0.3794	0.3729	0.3769
LSSH	5000	289.3	435.9	0.4527	0.4848	0.4688	0.3611	0.3510	0.3576
CMFH	5000	30.9	0.5	0.4348	0.3652	0.4000	0.3574	0.3483	0.3639
QCH	5000	586.6	0.4	0.5227	0.5064	0.5146	0.4341	0.4292	0.4312
STMH	5000	15.0	38.6	0.4374	0.5194	0.4784	0.3705	0.3549	0.3570
LSRH	5000	32.5	2.6	**0.5440**	**0.5398**	**0.5419**	**0.5132**	**0.5118**	**0.5125**

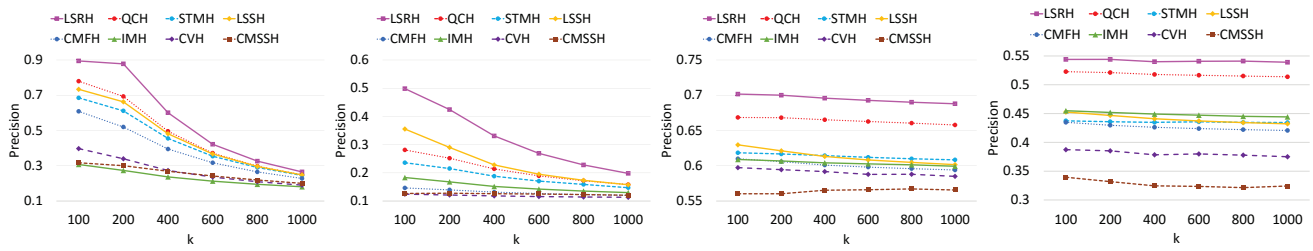
most of the datasets except for Wikipedia. Such observation is consistent with the results in previous research [11, 20, 21]. As explained in [11], there is a significant semantic gap between the two modalities of Wikipedia; the texts in Wikipedia are much better than the images in describing the semantic concept, thus leading to lower mAPs when images are used to query against the text database.

Another interesting finding is that the performance of LSRH always increases when the hash code becomes longer, while the performances of some of the baselines such as CVH, CMSSH and IMH do not increase or even slightly drop with longer codes. This has also been observed by [12, 34]. In fact, those methods are similar in the sense that they all solve certain eigen-decomposition problems with orthogonality constraints to reduce bit correlations. As a result, most discriminative information is contained within the first few bits. As the code becomes longer, the hash code will be gradually dominated by indiscriminative hash bits, which do not contribute to the retrieval performance.

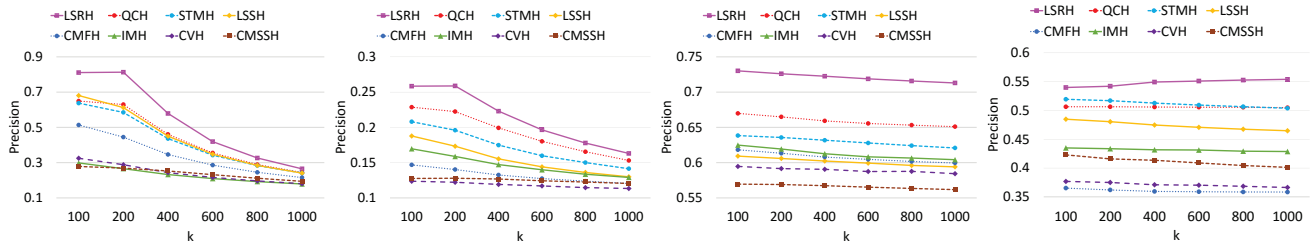
In addition to mAP, we also report the performances of different methods in terms of K-nearest neighbor precision and precision-recall. The results for those benchmarks on all the datasets are shown in Table 3, Figure 5 and Figure 6. Note that the precision-recall values in Table 3 are computed as the area under the precision-recall curves, and larger values mean better overall performance. From Table 3, we can observe that the relative performances of different methods are generally consistent with that of mAP. Specifically, LSRH consistently outperforms all the baselines across different datasets in both top-k precision and precision-recall, and the average performance gap between LSRH and the best baselines on the four datasets are 35%, 48%, 10% and 12% respectively.

In addition to the retrieval performance metrics, we have also shown the training and testing time for different algorithms under the same system settings in Table 3. The

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2016.2610969, IEEE Transactions on Pattern Analysis and Machine Intelligence

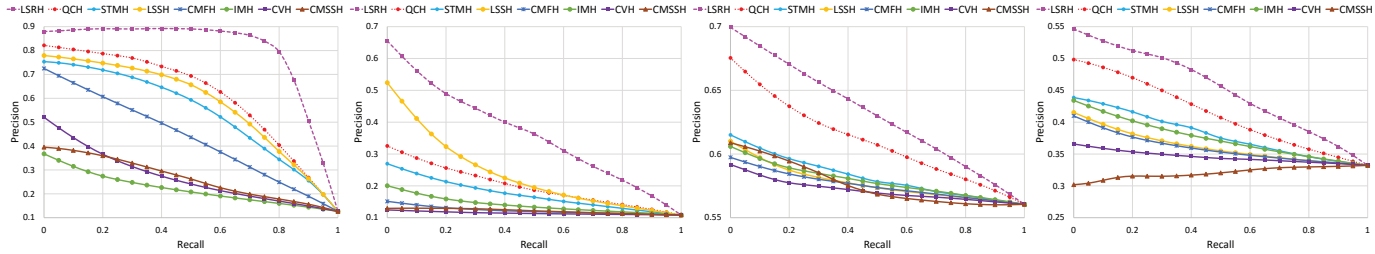


(a) Labelme (b) Wikipedia (c) MIRFlickr (d) NUSWIDE

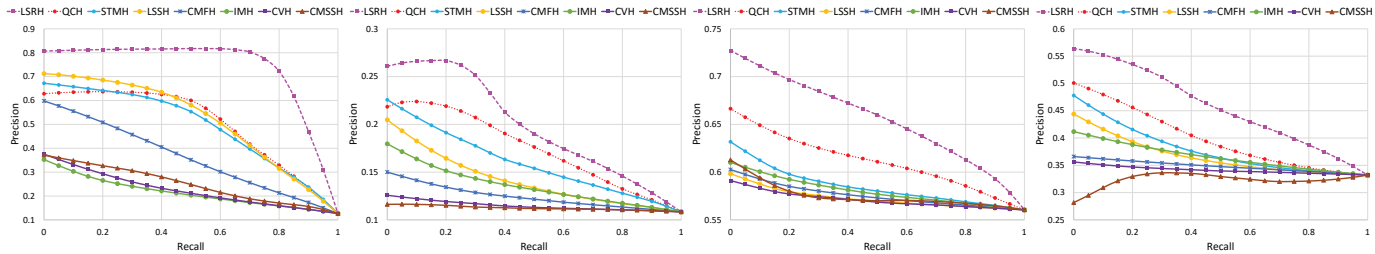


(e) Labelme (f) Wikipedia (g) MIRFlickr (h) NUSWIDE

Fig. 5: Top-k precision of all methods with 32-bit hash code and k varies from 100 to 1000. The first row shows the results of text-query-image and the second row shows that of image-query-text. The proposed LSRH outperforms all the baselines.



(a) Labelme (b) Wikipedia (c) MIRFlickr (d) NUSWIDE



(e) Labelme (f) Wikipedia (g) MIRFlickr (h) NUSWIDE

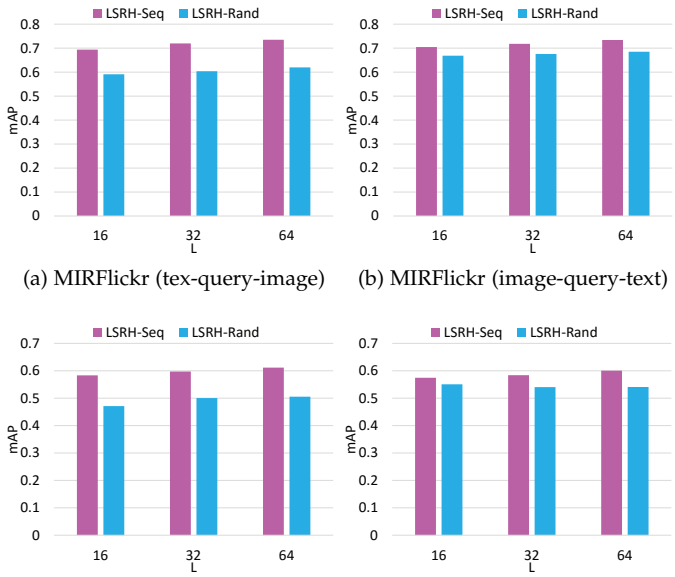
Fig. 6: Precision-recall curves with 32-bit hash code. The first row shows the results of text-query-image and the second row shows that of image-query-text. Larger area under the curve indicates better performance. LSRH achieves the best performance.

results are shown in seconds. We note that the proposed LSRH can be trained very fast compared to most of the baseline methods; while some of the competitive baselines such as LSSH and QCH take much longer to train. In terms of testing time, linear hashing algorithms are clear winners since the hash encoding stage only involves simple linear transformations followed by zero-thresholding. LSRH also falls into the linear hashing category since the ranking operation is performed upon linear projections. The close testing time of LSRH compared to most of the other methods confirms the efficiency of ranking-based hash encoding. We note that LSSH and STMH are two exceptions in the experiments with much longer testing time. This is caused by large-matrix inverse computations or nonlinear transformations in the hash encoding step, thus making them less effective in real-world applications that require online hash encoding. The moderate training and testing time further confirms the effectiveness of the proposed LSRH.

Overall, the proposed LSRH consistently achieves superior performance against the baselines in different metrics and datasets, with only moderate training and testing time. Such good performance of LSRH can be attributed to several reasons. Firstly, the ranking-based hash functions can be very useful in preserving the cross-modal similarities.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2016.2610969, IEEE Transactions on Pattern Analysis and Machine Intelligence

Second, the proposed hash learning procedures are both efficient and effective in learning the ranking-based hash functions. Third, the ranking-structure of cross-modal data exploited by the proposed hashing framework is very useful in bridging the semantic gap between different modalities.



(a) MIRFlickr (tex-query-image) (b) MIRFlickr (image-query-text)

(c) NUSWIDE (tex-query-image) (d) NUSWIDE (image-query-text)

Fig. 7: Comparison of different strategies in generating multiple hash codes. The results are obtained with 32-bit hash code on MIRFlickr and NUSWIDE.

4.6 Effect of sequential learning

In this section, we study the role of boosting in learning multiple hash codes. Specifically, we consider a randomized version of LSRH, denoted as LSRH-Rand, where each hash code is learned independently with random initialization. To differentiate from LSRH-Rand, we denote the sequential version of LSRH as LSRH-Seq. The results of this experiment are shown in Figure 7. We note that LSRH-Seq demonstrates consistent performance boost over LSRH-Rand, with approximately 10% lead on average. The performance gap is mainly due to the code redundancy in LSRH-Rand. Specifically, in LSRH-Rand, multiple independent random initializations may correspond to the same local minima of the objective function; as a result, there exist redundant hash functions that compromise the amount of discriminative information contained in the resultant hash codes. On the other hand, LSRH-Seq learns each new code by taking advantage of the information from previous ones and focus on a different subset of the training data. Therefore, LSRH-Seq codes contain more discriminative information than LSRH-Rand codes of the same length.

4.7 Different loss functions

The proposed ranking-hash framework is able to incorporate different types of loss functions with minimal modification to the hash learning procedures. Here we compare the the performance of four different loss functions. Specifically, the loss functions included in this experiment are L1, L2, exponential and hinge loss. Details of those loss functions have been explained in Section 3.8. The results are illustrated in Table 4 and Figure 8. We observe that the default L1 loss function usually achieves the best performance. This may be explained by the fact that the L1 loss directly follows from the empirical loss, which is closely related to the performance metrics used in the similarity search. On the other hand, the performance values of different loss functions are very close in most test cases. The slight differences in the performances are caused by different ways of assigning penalties to true or false similarity predictions. In general, the ability to accommodate different loss functions in a unified hash learning framework greatly extends the flexibility of LSRH.

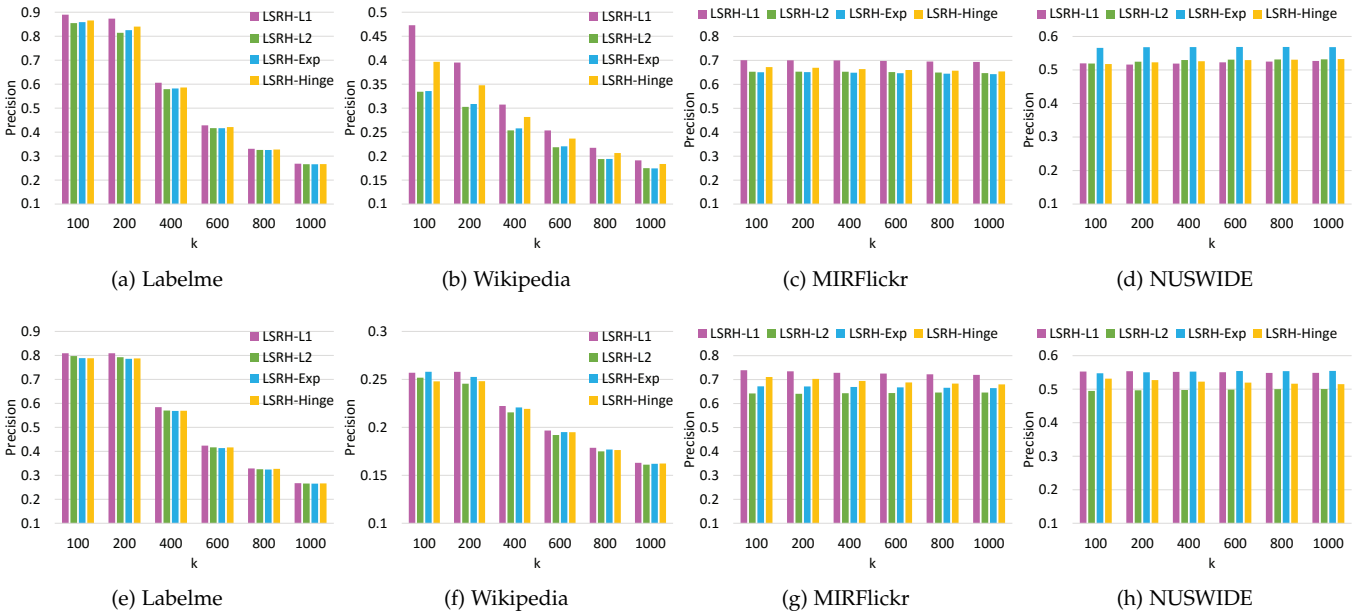
TABLE 4: mAP and top-100 precision of LSRH using different loss functions. The hash code length is set to 32 bits

	Text query image		Image query text	
Loss function	mAP	Precision	mAP	Precision
		Labelme		
LSRH-L1	**0.8931**	**0.8898**	**0.8167**	**0.8108**
LSRH-L2	0.8664	0.8547	0.8136	0.7806
LSRH-Exp	0.8569	0.8587	0.8005	0.7854
LSRH-Hinge	0.8597	0.8656	0.8027	0.7981
		Wikipedia		
LSRH-L1	**0.6168**	**0.4730**	**0.2849**	0.2569
LSRH-L2	0.4360	0.3343	0.2813	0.2518
LSRH-Exp	0.4367	0.3357	0.2874	**0.2580**
LSRH-Hinge	0.5292	0.3966	0.2771	0.2479
		MIRFlickr		
LSRH-L1	**0.7230**	**0.7008**	**0.7680**	**0.7392**
LSRH-L2	0.6774	0.6526	0.6712	0.6421
LSRH-Exp	0.6730	0.6506	0.6963	0.6718
LSRH-Hinge	0.6995	0.6721	0.7474	0.7120
		NUSWIDE		
LSRH-L1	**0.5855**	**0.5525**	0.5483	0.5198
LSRH-L2	0.5314	0.4948	0.5412	0.5193
LSRH-Exp	0.5777	0.5474	**0.5997**	**0.5660**
LSRH-Hinge	0.5849	0.5315	0.5507	0.5177

4.8 Effect of subspace dimension

Here we study the performance of LSRH with different subspace dimension K. In this experiment, we vary K from 2 up to 32 in linear scale of $\log_2 K$ (i.e. $K = 2^1, \cdots, 2^5$). Note that the choice of K is not restricted to the powers of two, and the reason for our settings here is to make sure that there's a bijective mapping from each K-ary code to a binary code with the same number of bits. Recall that we train $\lfloor N_b / \lceil \log_2 K \rceil \rfloor$ LSRH codes when evaluating the performance at N_b bits. Here we choose N_b to be 60 bits since it is the common multiple of 1 to 5. This way, we can make sure that the same amount of information is contained in each LSRH codeword irrespective of the choice of K. The results of this experiment are shown in Figure 9. We note that the effect of K is different on different datasets and retrieval tasks. For example, the performance of both retrieval tasks on Labelme doesn't change much when K varies from 4 to 32; while on MIRFlickr, the performance of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2016.2610969, IEEE Transactions on Pattern Analysis and Machine Intelligence

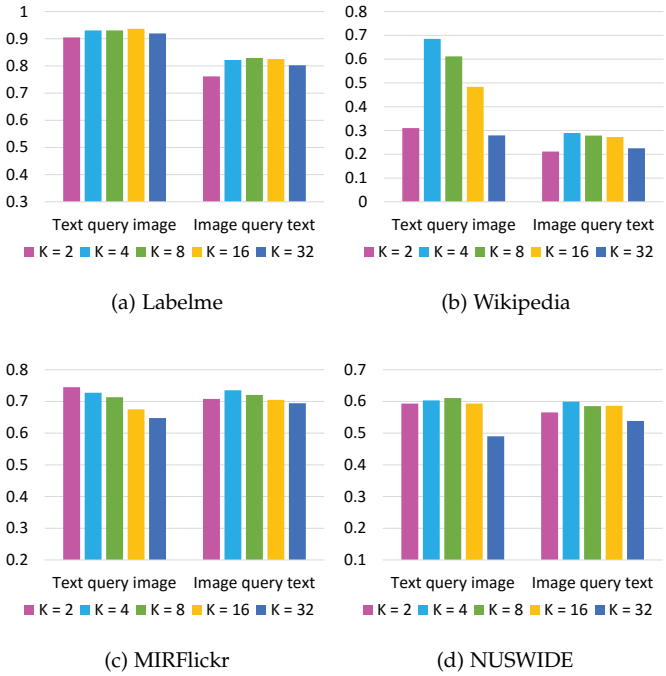


(a) Labelme (b) Wikipedia (c) MIRFlickr (d) NUSWIDE

(e) Labelme (f) Wikipedia (g) MIRFlickr (h) NUSWIDE

Fig. 8: Top-k precision of LSRH trained with different loss functions. The length of the hash code is set to 32 bits and k varies from 100 to 1000. The first and second rows show the results of text-query-image and image-query-text respectively.

text-query-image slightly decreases with the increase of K. Such distinctions indicate that the ranking-structures inherent in different datasets are different and the best practice is to use cross-validation to choose the optimal subspace dimension. Overall, we find that $K = 4$ is an all-around good choice across different benchmarks and datasets.



(a) Labelme (b) Wikipedia

(c) MIRFlickr (d) NUSWIDE

Fig. 9: The mAP with 60-bit hash code under different choices of subspace dimensions. K = 4 leads to the best overall performance.

4.9 Scalability Study

In order to test the scalability of the proposed hash learning method, we profile its training time under varying training sizes and compare with the baseline methods. Specifically, we vary the training size from 10^6 to 10^8 pairs on two of the larger datasets: MIRFlickr and NUSWIDE. All of the profiled algorithms run on the same system with Intel Xeon E5-2680 CPU @ 2.5 GHz and 128 GB of memory. The results of this test are summarized in Table 5. Note that all of the compared algorithms are implemented using MATLAB, and are therefore comparable under the same test settings. We can observe that LSRH can be trained significantly faster than the most competitive baselines such as LSSH and QCH. Additionally, the training time of LSRH only increases moderately with the increase in training size. The short training time and good scalability with large training sets demonstrate the effectiveness of our learning procedures.

TABLE 5: Training time of 32-bit hash code on MIRFlickr and NUSWIDE. The training size is varied from 10^6 to 10^8 pairs and the results are in seconds.

	Training size (pairs)					
Method	10^6	10^7	10^8	10^6	10^7	10^8
	MIRFlickr			NUSWIDE		
CVH	0.04	0.04	0.07	0.8	1.1	1.1
CMSSH	197.8	219.2	220.1	508.6	504.9	508.6
IMH	0.5	5.0	119.4	0.6	5.3	119.7
LSSH	536.7	415.7	610.6	383.0	396.7	711.7
CMFH	1.2	6.2	28.6	3.1	10.6	53.8
QCH	46.8	134.5	366.1	89.4	234.0	603.3
STMH	6.8	16.1	36.5	10.9	26.0	66.6
LSRH	4.5	9.6	33.2	6.7	11.6	34.7

5 Conclusion and Future Work

In this paper, we propose a novel cross-modal hashing method, referred to as Linear Subspace Ranking Hashing (LSRH), for large-scale cross-modal similarity search. Specifically, we exploit a new class of hash functions based on the ranking structure of feature subspaces. For hash function learning, we develop an effective ranking-based hash learning framework that can flexibly accommodate different loss functions with minimal changes to the learning procedures. The overall learning procedure leads to two sets of optimal ranking subspaces that maximally preserve the cross-modal similarity. We conduct extensive experiments on widely used multimodal datasets and compare with a range of state-of-the-art cross-modal hashing methods. The experimental results demonstrate the superiority of LSRH in generating highly discriminative compact hash codes for cross-modal retrieval tasks.

Our future work consists of two directions to further push the limits of ranking-based hashing. The first direction is to study the ranking structure of nonlinear subspaces (e.g. kernel space), which may potentially reveal more discriminative ranking structure that cannot be discovered by linear subspaces. The second direction involves extending the single-layer ranking function to a multi-layer end-to-end deep network which incorporates feature learning and ranking encoding in a seamless deep learning architecture.

Acknowledgement

This material is based upon work partially supported by the NASA under Grant Number NNX15AV40A. Any opinions, findings, and conclusion or recommendations expressed in this materials are those of the authors and do not necessarily reflect the views of NASA.

References

[1] K. Grauman and R. Fergus, "Learning binary hash codes for large-scale image search," in *Machine learning for computer vision*. Springer, 2013, pp. 49–87.

[2] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the twentieth annual symposium on Computational geometry*. ACM, 2004, pp. 253–262.

[3] M. Norouzi and D. J. Fleet, "Minimal loss hashing for compact binary codes," in *ICML, 2011)*, 2011, pp. 353–360.

[4] Y. Gong and S. Lazebnik, "Iterative quantization: A procrustean approach to learning binary codes," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 817–824.

[5] G. Lin, C. Shen, Q. Shi, A. van den Hengel, and D. Suter, "Fast supervised hashing with decision trees for high-dimensional data," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*. IEEE, 2014, pp. 1971–1978.

[6] W.-C. Kang, W.-J. Li, and Z.-H. Zhou, "Column sampling based discrete supervised hashing," 2016.

[7] F. Shen, C. Shen, W. Liu, and H. Tao Shen, "Supervised discrete hashing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 37–45.

[8] M. M. Bronstein, A. M. Bronstein, F. Michel, and N. Paragios, "Data fusion through cross-modality metric learning using similarity-sensitive hashing," in *CVPR, 2010*, pp. 3594–3601.

[9] Y. Zhen and D.-Y. Yeung, "Co-regularized hashing for multimodal data," in *NIPS, 2012*, pp. 1376–1384.

[10] J. Song, Y. Yang, Y. Yang, Z. Huang, and H. T. Shen, "Inter-media hashing for large-scale retrieval from heterogeneous data sources," in *ACM SIGMOD, 2013*, pp. 785–796.

[11] J. Zhou, G. Ding, and Y. Guo, "Latent semantic sparse hashing for cross-modal similarity search," in *ACM SIGIR, 2014*, pp. 415–424.

[12] G. Ding, Y. Guo, and J. Zhou, "Collective matrix factorization hashing for multimodal data," in *CVPR, 2014*, pp. 2083–2090.

[13] K. Li, J. Ye, and K. A. Hua, "What's making that sound?" in *Proceedings of the 22Nd ACM International Conference on Multimedia*, ser. MM '14. ACM, 2014, pp. 147–156.

[14] Y. Zhuang, Z. Yu, W. Wang, F. Wu, S. Tang, and J. Shao, "Cross-media hashing with neural networks," in *ACM MM, 2014*, pp. 901–904.

[15] 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*. ACM, 2011, pp. 225–234.

[16] J. Masci, M. M. Bronstein, A. M. Bronstein, and J. Schmidhuber, "Multimodal similarity-preserving hashing," *TPAMI, 2014*.

[17] M. Ou, P. Cui, F. Wang, J. Wang, W. Zhu, and S. Yang, "Comparing apples to oranges: a scalable solution with heterogeneous hashing," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2013, pp. 230–238.

[18] Z. Lin, G. Ding, M. Hu, and J. Wang, "Semantics-preserving hashing for cross-view retrieval," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3864–3872.

[19] S. Moran and V. Lavrenko, "Regularised cross-modal hashing," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '15. ACM, 2015, pp. 907–910.

[20] B. Wu, Q. Yang, W.-S. Zheng, Y. Wang, and J. Wang, "Quantized correlation hashing for fast cross-modal search," in *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI*, 2015.

[21] D. Wang, X. Gao, X. Wang, and L. He, "Semantic topic multimodal hashing for cross-media retrieval," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 2015, pp. 3890–3896.

[22] M. Melucci, "On rank correlation in information retrieval evaluation," in *ACM SIGIR Forum*, vol. 41, no. 1. ACM, 2007, pp. 18–33.

[23] J. Yagnik, D. Strelow, D. A. Ross, and R.-s. Lin, "The power of comparative reasoning," in *ICCV, 2011*, pp. 2431–2438.

[24] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher, "Min-wise independent permutations," *Journal of Computer and System Sciences*, vol. 60, no. 3, pp. 630 – 659, 2000.

[25] K. Li, G. Qi, J. Ye, and K. Hua, "Cross-modal hashing through ranking subspace learning," in *IEEE International Conference on Multimedia and Expo, ICME*, 2016.

[26] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, "Supervised hashing with kernels," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 2074–2081.

[27] J. Wang, J. Wang, N. Yu, and S. Li, "Order preserving hashing for approximate nearest neighbor search," in *Proceedings of the 21st ACM international conference on Multimedia*. ACM, 2013, pp. 133–142.

[28] S. Kumar and R. Udupa, "Learning hash functions for cross-view similarity search," in *IJCAI, 2011*, vol. 22, no. 1, p. 1360.

[29] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Advances in neural information processing systems*, 2009, pp. 1753–1760.

[30] N. Quadrianto and C. H. Lampert, "Learning multi-view neighborhood preserving projections," in *ICML, 2011*, pp. 425–432.

[31] Y. Zhen and D.-Y. Yeung, "A probabilistic model for multimodal hash function learning," in *SIGKDD, 2012*.

[32] D. Zhai, H. Chang, Y. Zhen, X. Liu, X. Chen, and W. Gao, "Parametric local multimodal hashing for cross-view similarity search," in *IJCAI, 2013*, pp. 2754–2760.

[33] X. Zhu, Z. Huang, H. T. Shen, and X. Zhao, "Linear cross-modal hashing for efficient multimedia search," in *ACM MM, 2013*, pp. 143–152.

[34] D. Zhang and W.-J. Li, "Large-scale supervised multimodal hashing with semantic correlation maximization," in *AAAI, 2014*.

[35] F. Wu, Z. Yu, Y. Yang, S. Tang, Y. Zhang, and Y. Zhuang, "Sparse multi-modal hashing," *IEEE TMM, 2014*.

[36] H. L. Liu, J. Rongrong, W. Yongjian, and H. Gang, "Supervised matrix factorization for cross-modality hashing," in *IJCAI, 2016*.

[37] T. Zhang and J. Wang, "Collaborative quantization for cross-modal similarity search."

[38] G. Irie, H. Arai, and Y. Taniguchi, "Alternating co-quantization for cross-modal hashing," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1886–1894.

[39] D. Wang, P. Cui, M. Ou, and W. Zhu, "Deep multimodal hashing with orthogonal regularization," in *Proceedings of the 24th International Conference on Artificial Intelligence*. AAAI Press, 2015, pp. 2291–2297.

[40] Y. Cao, M. Long, J. Wang, and H. Zhu, "Correlation autoencoder hashing for supervised cross-modal search," in *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*. ACM, 2016, pp. 197–204.

[41] Y. Cao, M. Long, W. Jianmin, Q. Yang, and P. Yu, "Deep visual-semantic hashing for cross-modal retrieval," in *SIGKDD, 2016*.

[42] Q.-Y. Jiang and W.-J. Li, "Deep cross-modal hashing," *arXiv preprint arXiv:1602.02255*, 2016.

[43] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

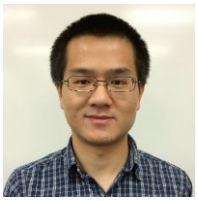
[44] N. Rasiwasia, J. Costa Pereira, E. Coviello, G. Doyle, G. Lanckriet, R. Levy, and N. Vasconcelos, "A New Approach to Cross-Modal Multimedia Retrieval," in *ACM MM, 2010*, 2010, pp. 251–260.

[45] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *International journal of computer vision*, vol. 42, no. 3, pp. 145–175, 2001.

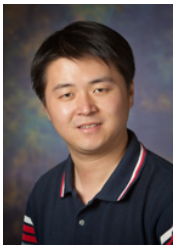
[46] M. J. Huiskes and M. S. Lew, "The mir flickr retrieval evaluation," in *MIR '08: Proceedings of the 2008 ACM International Conference on Multimedia Information Retrieval*. New York, NY, USA: ACM, 2008.

[47] T.-S. 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*, ser. CIVR '09. New York, NY, USA: ACM, 2009, pp. 48:1–48:9.

Kai Li received the B.S. degree in Automation from Huazhong University of Science and Technology, Wuhan, China, in 2010. He is currently working toward the PhD degree in the Department of Computer Science, University of Central Florida. His research interests include information retrieval, machine learning, multimedia, and computer vision and pattern recognition. His current focus is on multimedia hashing for large-scale similarity search.



Jun Ye holds a M.S. degree in Computer Science from Beihang University, Beijing, China, in 2010. He is currently working toward the PhD degree in the Department of Computer Science, University of Central Florida. His research interests include multimedia retrieval, multimedia data analysis and machine learning. His current focus is on human-centric live video computing and human action recognition.



Guojun Qi received the PhD degree from the University of Illinois at Urbana-Champaign, in December 2013. His research interests include pattern recognition, machine learning, computer vision, multimedia, and data mining. He received twice IBM PhD fellowships, and Microsoft fellowship. He is the recipient of the Best Paper Award at the 15th ACM International Conference on Multimedia, Augsburg, Germany, 2007. He is currently a faculty member with the Department of Electrical Engineering and Computer Science at the University of Central Florida, and has served as program committee member and reviewer for many academic conferences and journals in the fields of pattern recognition, machine learning, data mining, computer vision, and multimedia.

Kien A. Hua is a Pegasus Professor in the Computer Science Department at University of Central Florida, and he is the Director of the Data Systems Lab. He served as the Associate Dean for Research of the College of Engineering and Computer Science at UCF. Prior to joining the university, he was with IBM. Dr.Hua received his B.S. in Computer Science, and M.S. and Ph.D. in Electrical Engineering, all from the University of Illinois at Urbana-Champaign. His diverse expertise includes network and wireless communications, image/video computing, sensor networks, medical imaging, databases, mobile computing, and intelligent transportation systems. He has published widely, with over 10 papers recognized as best/top papers at conferences and a journal. Dr. Hua has served as a Conference Chair, an Associate Chair, and a Technical Program Committee Member of numerous international conferences, as well as on the editorial boards of a number of professional journals. Dr. Hua is a Fellow of IEEE.