High-Dimensional Sparse Cross-Modal Hashing with Fine-Grained Similarity Embedding

Yongxin Wang
School of Software, Shandong University
Jinan, China
yxinwang@hotmail.com

Zhen-Duo Chen
School of Software, Shandong University
Jinan, China
chenzd.sdu@gmail.com

Xin Luo*
School of Software, Shandong University
Jinan, China
luoxin.lxin@gmail.com

Xin-Shun Xu
School of Software, Shandong University
Jinan, China
xuxinshun@sdu.edu.cn

ABSTRACT

Recently, with the discoveries in neurobiology, high-dimensional sparse hashing has attracted increasing attention. In contrast with general hashing that generates low-dimensional hash codes, the high-dimensional sparse hashing maps inputs into a higher dimensional space and generates sparse hash codes, achieving superior performance. However, the sparse hashing has not been fully studied in hashing literature yet. For example, how to fully explore the power of sparse coding in cross-modal retrieval tasks; how to discretely solve the binary and sparse constraints so as to avoid the quantization error problem. Motivated by these issues, in this paper, we present an efficient sparse hashing method, i.e., High-dimensional Sparse Cross-modal Hashing, HSCH for short. It not only takes the high-level semantic similarity of data into consideration, but also properly exploits the low-level feature similarity. In specific, we theoretically design a fine-grained similarity with two critical fusion rules. Then we take advantage of sparse codes to embed the fine-grained similarity into the to-be-learnt hash codes. Moreover, an efficient discrete optimization algorithm is proposed to solve the binary and sparse constraints, reducing the quantization error. In light of this, it becomes much more trainable, and the learnt hash codes are more discriminative. More importantly, the retrieval complexity of HSCH is as efficient as general hash methods. Extensive experiments on three widely-used datasets demonstrate the superior performance of HSCH compared with several state-of-the-art cross-modal hashing approaches.

CCS CONCEPTS

• **Computing methodologies** → **Learning paradigms**; • **Information systems** → **Multimedia and multimodal retrieval**.

*Corresponding Author

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3449798

KEYWORDS

Sparse hashing; high-dimensional hashing; cross-modal hashing; discrete optimization

ACM Reference Format:
Yongxin Wang, Zhen-Duo Chen, Xin Luo, and Xin-Shun Xu. 2021. High-Dimensional Sparse Cross-Modal Hashing with Fine-Grained Similarity Embedding. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3442381.3449798

1 INTRODUCTION

With the explosive growth of multimedia data on the Internet, information retrieval tends to focus on large-scale and high-dimensional problems. The commonly used nearest neighbor (NN) search becomes impractical due to its expensive computational cost [52]. To accelerate retrieval speed and reduce storage cost, approximate nearest neighbor (ANN) search via hashing is proposed and popularly used to seek a balance between speed and accuracy. It maps high-dimensional features into a low-dimensional Hamming space; In this way, the search becomes much faster in the Hamming space and the storage cost is also significantly reduced.

Very recently, with biological observations on the fruit fly's olfactory circuit, some studies have shown the mechanism of high-dimensional sparse coding [8, 25, 35, 40]. From the perspective of computer science, the fly's circuit can be viewed as a hash function. In detail, it maps each sample to a higher-dimensional Hamming space by a sparse binary mapping matrix; then each sample is represented as a sparse binary hash code after a winner-take-all (WTA) competition [26, 44]. Compared with traditional hashing methods, high-dimensional sparse coding has two advantages: 1) Sparse codes have more powerful representation capability. For instance, an r-bit general hash code only has 2^r different representations; while a k-dimensional sparse code with r ones and other zeros has C_k^r different combinations of expressions. Usually, k is very large, and $C_k^r \gg 2^r$. More expressions mean better preservation of similarity information and correspondingly better accuracy. 2) The computational cost of calculating Hamming distance for sparse codes is only relevant to the number of nonzero elements, i.e., $O(r)$ for a sparse code with r ones, which is the same as an r-bit dense hash code. Therefore, when we pursue the same accuracy, sparse codes can greatly improve the query speed because the number of

nonzero elements required in sparse codes is much smaller than the code length of dense hash codes.

However, despite the powerful capability of sparse coding, it has not been fully studied in hashing literature yet. For example, some methods optimize the sparse constraint via ℓ_0 or ℓ_1-norm, which usually causes NP-hard optimization problems [25]. In addition, some methods firstly solve the optimization to generate a real-valued intermediate variable and then binarize it by a WTA operation, resulting in a large quantization error [40]. More importantly, to the best of our knowledge, all existing sparse hashing methods are designed for unimodal retrieval tasks, in which both the query and retrieval samples are with the same modality. However, in real-world retrieval, data usually consists of multiple modalities. Correspondingly, there is an increasing demand for cross-modal retrieval, e.g., using images to search a database with other modalities, such as text or audio. If we leverage sparse codes in cross-modal retrieval tasks, it is much possible to bridge the heterogeneous gap between different modalities because sparse codes are able to generate more common Hamming space for different modalities.

From another point of view, the data information is usually under-explored in hashing literature. For example, besides $O(n^2)$ computation and storage cost, the widely-used binary pairwise similarity may lose a lot of fine-grained similar information. It treats a pair as similar if they share at least one common label [19, 28, 31, 37]. Actually, a pair with more than one common label should be more similar than a pair with only one common label, rather than simply treating them as the same. More importantly, the low-level feature similarity is less taken into consideration, which is extremely important for fine-grained retrieval [12]. Nevertheless, simply fusing different levels of similarity, i.e., the high-level semantic similarity and low-level feature similarity, may inevitably lead to conflicts between different levels due to the "semantic gap problem". We argue that an ideal similarity should take both the high-level and low-level similarities into consideration according to some well-designed rules. For example, the high-level semantic similarity should always take precedence over the low-level feature similarity; because numerous studies have shown that the supervised information, as the most consistent information of different modalities, is a guarantee of good accuracy. Meanwhile, a more fine-grained similarity also puts higher requirements on the representation ability of hash codes. Naturally, high-dimensional sparse coding could meet this demand properly. If the power of sparse representation is fully explored, the learnt hash codes are capable of carrying more data information, and subsequently generate more fine-grained retrieval results.

Motivated by the above issues, in this paper, we present a novel high-dimensional sparse hashing method for supervised cross-modal retrieval, named as High-dimensional Sparse Cross-modal Hashing, HSCH for short. Specifically, it computes a fine-grained similarity matrix, which theoretically combines high-level semantic similarity and low-level feature similarity. Then it exploits the power of high-dimensional sparse coding to embed the fine-grained similarity into hash codes.

To summarize, the main contributions of this paper are described as follows:

- We present a novel high-dimensional sparse cross-modal hashing method. It makes use of the advantages of high-dimensional sparse coding to generate powerful sparse hash codes, thus resulting in more fine-grained retrieval results. To the best of our knowledge, it is the first high-dimensional sparse hashing method for cross-modal retrieval.
- It takes both the high-level semantic similarity and low-level feature similarity into consideration, and generates a fine-grained similarity with two well-designed fusion rules. With the help of sparse codes, the fine-grained similarity could be well embedded into hash codes.
- We also propose an efficient discrete optimization algorithm, which iteratively settles the binary and sparse constraints of hash codes, avoiding the large quantization error problem.
- Extensive experimental results conducted on three widely-used benchmarks verify the effectiveness of our HSCH in terms of accuracy and efficiency.

The rest of this paper is organized as follows. In Section 2, we briefly review some related dense and sparse hashing methods. Section 3 gives details of our HSCH. Section 4 reports the experimental results and detailed analyses. Finally, Section 5 concludes the paper.

2 RELATED WORK

2.1 Dense Hashing

Generally speaking, without special statements, hashing refers to *dense hashing*. It maps input data into low-dimensional binary hash codes and applies the mapping matrix as a hash function. Typical examples include Iterative Quantization (ITQ) [13], Discrete Graph Hashing (DGH) [30], Discrete Locality Linear Embedding (DLLH) [18], Supervised Hashing with Kernels (KSH) [31], Supervised Discrete Hashing (SDH) [41], Discrete Hashing with Multiple Supervision (MSDH) [34], Deep Semantic Reconstruction Hashing (DSRH) [51], and so on. Most pioneering hashing models focus on the unimodal retrieval task.

To further deal with the cross-modal retrieval tasks, many cross-modal hashing methods have also been proposed and attracted increasing attention [50]. According to whether supervised information is used, they can be classified into unsupervised and supervised ones. Unsupervised ones learn hash functions only from the data without any supervised information [45]. Representative methods include Inter-Media Hashing (IMH) [43], Collective Matrix Factorization Hashing (CMFH) [9], Composite Correlation Quantization (CCQ) [32], Fusion Similarity Hashing (FSH) [29], Robust and Flexible Discrete Hashing (RFDH) [47], Collective Reconstructive Embedding (CRE) [16], Robust Unsupervised Cross-Modal Hashing (RUCMH) [5], Unsupervised Knowledge Distillation (UKD) [15], etc.

Due to the fact that different modalities usually have large heterogeneous gaps, unsupervised cross-modal hashing cannot achieve satisfying performance [37]. Hence, the supervised information, as the most consistent information of different modalities, becomes a guarantee of good accuracy. Consequently, many supervised cross-modal hashing methods have been proposed. For instance, Discriminative Cross-modal Hashing (DCH) [54] transforms the hash learning into a classification problem and solves the binary constraints via a discrete cyclic coordinate descent (DCC) approach.

Scalable Discrete Matrix Factorization Hashing (SCRATCH) [3] learns a common subspace of labels and multiple features through collective matrix factorization. Label Consistent Matrix Factorization Hashing (LCMFH) [46] directly utilizes the multi-modal data and leverages labels to guide the hash learning step by label consistent matrix factorization. Discrete Latent Factor Hashing (DLFH) [20] maximizes the log-likelihood between pairwise similarity and hash codes of different modalities, then discretely solves the binary constrained optimization problem without relaxation.

In general, dense hashing is limited by its weak representation ability. To obtain satisfactory accuracy, longer code lengths are often required, leading to greater storage and query costs.

2.2 Sparse Hashing

It is worth noting that, *sparse hashing* in this paper means high-dimensional sparse binary coding, rather than applying sparse constraints of dictionary learning in dense hashing [23, 48, 55]. In addition, hash codes in dense hashing are dense binary codes, while that in sparse hashing are sparse binary ones.

Considering the advantages of sparse codes on accuracy and query speed, some sparse hashing methods have been proposed. The first work FlyHash [8] generalizes the insight from the fly's circuit to develop a sparse Locality Sensitivity Hashing (LSH) [2]. Specifically, it maps each input sample into a higher-dimensional vector by multiplying a randomly generated sparse binary matrix. Then a WTA operation is applied to achieve sparsification with only the top-r elements being set to one and all others being zero. It is obvious that, like LSH, FlyHash is also a data-independent hash method that does not consider rich data information. Intuitively, a more powerful mapping matrix learnt from data might help to improve the performance of FlyHash.

Therefore, some data-dependent sparse hashing methods have been proposed. Thereinto, Optimal Sparse Lifting Hashing (OSLHash) [25] is inspired by FlyHash, which attempts to inject the data into sparse hash learning, demonstrating better performance than FlyHash. In specific, it first computes the optimal sparse lifting of data, and then learns the optimal lifting operator to map data into the optimal sparse lifting. Besides, it introduces the ℓ_p pseudo-norm to optimize sparsity and binarization constraints. Nevertheless, OSLHash has to solve a constrained linear program, which is very time-consuming. Compared with OSLHash, Sparse Binary Projections (SBP) [35] learns sparse binary codes and sparse binary projections by an alternative optimization algorithm; thus, the efficiency of SBP is significantly improved. However, it designs a simple learning scheme, which might not fully consider the data information. Most recently, Bio-inspired Hashing (BioHash) [40] adopts a biologically plausible unsupervised approach [22] to compute the sparse projections and then applies the WTA competition to realize binarization. Although it biologically learns the hash functions, the independent WTA step brings a large quantization error.

In general, none of the existing sparse methods are designed for supervised cross-modal retrieval tasks. Therefore, in this paper, HSCH is designed for cross-modal retrieval; at the same time, it focuses on how to leverage the powerful sparse coding to maintain a fine-grained similarity of data and accordingly seek more fine-grained retrieval results at a practicable computational cost.

3 METHOD

3.1 Notations

Let $\mathbf{X}^{(l)} \in \mathfrak{R}^{d_l \times n} (l \in \{1, 2, ..., m\})$ denote the training set of the l-th modality, where n is the number of samples, m is the number of modalities, and d_l is the feature dimensionality. $\mathbf{L} \in \mathfrak{R}^{c \times n}$ is the ground-truth label matrix, where c is the number of categories. $\mathbf{L}_{ij} = 1$ if the j-th instance belongs to the i-th category, and 0 otherwise. $\mathbf{B} \in \{0, 1\}^{k \times n}$ is the to-be-learnt high-dimensional sparse hash codes, with each column involving only r ones and all others being zero. k is the dimensionality of sparse codes, which is usually very high. Let the number of nonzero elements, i.e., r, denote the code length of sparse codes. Note that for dense hashing, the code length of a dense code is equal to its code dimensionality, i.e., r. $\mathrm{tr}(\cdot)$ is the trace operator. $|\cdot|$ is the absolute value function. $\|\cdot\|$ indicates the ℓ_2-norm for a vector or Frobenius-norm for a matrix. $\mathbf{I}$, $\mathbf{1}$, and $\mathbf{0}$ denote an identify matrix, an all one vector, and an all zero vector, respectively.

3.2 Hash Codes Learning

3.2.1 Fine-Grained Sparse Coding. To learn high-quality hash codes, we expect to embed the similarity of data as much as possible into high-dimensional sparse hash codes. Mathematically, we adopt the inner product minimization objective as follows:

$$\begin{aligned} &\min_{\mathbf{B}} \|\mathbf{B}^\top \mathbf{B} - r\mathbf{S}\|^2, \\ &s.t.\ \mathbf{B} \in \{0, 1\}^{k \times n},\ \mathbf{B}^\top \mathbf{1}_k = r\mathbf{1}_n. \end{aligned} \tag{1}$$

Here, the first binary constraint on $\mathbf{B}$ restricts its elements to 0 or 1; the second sparse constraint makes its each column contain r ones. $\mathbf{S}$ is a pairwise similarity. Usually, $\mathbf{S}_{ij} = 1$ if the i-th and j-th instances share at least one common label, and -1 otherwise [31]. This binary similarity discards a large amount of semantic information, especially for multi-label data, which limits the powerful representation ability of sparse coding. From another point of view, the low-level feature similarity is completely neglected. We argue that for samples with the same high-level semantic similarity value, they should be further ordered according to their low-level feature similarity values. Therefore, the role of low-level feature is non-negligible for seeking fine-grained retrieval results.

To solve the above issues, we first formulate the cosine similarity of each level as follows:

$$\mathbf{A}^{(s)} = \bar{\mathbf{L}}^\top \bar{\mathbf{L}}, \quad \mathbf{A}^{(f)} = \bar{\mathbf{X}}^\top \bar{\mathbf{X}}, \tag{2}$$

where $\mathbf{A}^{(s)}$ and $\mathbf{A}^{(f)}$ are the high-level semantic similarity and low-level feature similarity, respectively. $\bar{\mathbf{L}}$ is the ℓ_2-norm column normalized label matrix, with its i-th column defined as $\bar{\mathbf{L}}_i = \mathbf{L}_i / \|\mathbf{L}_i\|$. We temporarily use $\bar{\mathbf{X}}$ here to denote an ℓ_2-norm column normalized feature matrix, and we will specify it for different modalities afterwards.

To learn a high-level and low-level fused similarity, we first define a fused affinity as follows:

$$\mathbf{A} = a\mathbf{A}^{(s)} + b\mathbf{A}^{(f)}, \tag{3}$$

where a and b are parameters to balance the importance of different levels. To generate a high-quality fused affinity, we design two critical fusion rules:

- **Multi-modal.** Since data consists of multiple modalities, the low-level feature similarity $\mathbf{A}^{(f)}$ is calculated by all modalities. In detail, it is specified as follows:

$$\mathbf{A}^{(f)} = \sum_{l=1}^{m} \eta^{(l)} \bar{\mathbf{X}}^{(l)\top} \bar{\mathbf{X}}^{(l)}, \quad s.t. \sum_{l=1}^{m} \eta^{(l)} = 1, \tag{4}$$

  where $\eta^{(l)}$ is a balance parameter that is set to equal for all modalities, i.e., $\eta^{(l)} = 1/m$.
- **Prioritized.** As proved in hashing literature, supervised methods that exploit supervised information always achieve much better performance than unsupervised methods that only use low-level features, which indicates that the high-level supervised information is much more important than the low-level features. In other words, the high-level semantic similarity should always has a higher priority than the low-level feature similarity. Formally, for any $< i, j, k >$ samples, if $\mathbf{A}^{(s)}_{ij} > \mathbf{A}^{(s)}_{ik}$, it has to satisfy $\mathbf{A}_{ij} > \mathbf{A}_{ik}$, that is, $a\mathbf{A}^{(s)}_{ij} + b\mathbf{A}^{(f)}_{ij} > a\mathbf{A}^{(s)}_{ik} + b\mathbf{A}^{(f)}_{ik}$. After some mathematical transformations, we have $\frac{a}{b} > \big(\max(|\mathbf{A}^{(f)}_{ij} - \mathbf{A}^{(f)}_{ik}|)\big) / \big(\min(|\mathbf{A}^{(s)}_{ij} - \mathbf{A}^{(s)}_{ik}|)\big)$. According to the definitions of $\mathbf{A}^{(s)}$ and $\mathbf{A}^{(f)}$, it is easily to get $\max(|\mathbf{A}^{(f)}_{ij} - \mathbf{A}^{(f)}_{ik}|) = 1 + \sum_l g^{(l)}\eta^{(l)}$, where $g^{(l)} = 0$ if $\mathbf{X}^{(l)}$ is a nonnegative matrix, and 1 otherwise. $\min(|\mathbf{A}^{(s)}_{ij} - \mathbf{A}^{(s)}_{ik}|) = 1$ for single-label data, and $\min(|\mathbf{A}^{(s)}_{ij} - \mathbf{A}^{(s)}_{ik}|) = \frac{2}{\sqrt{c}\sqrt{c}} - \frac{2}{\sqrt{c}\sqrt{c+2}}$ for multi-label data, where c is the number of classes. Because we eventually use the normalized similarity, we set a and b as follows:

$$a = \begin{cases} \dfrac{p}{1} + \epsilon, & \text{single-label,} \\ \dfrac{pc(c+2) + pc\sqrt{c(c+2)}}{4} + \epsilon, & \text{multi-label,} \end{cases} \quad b = 1, \tag{5}$$

  where ϵ is a very small constant, and $p = 1 + \sum_l g^{(l)}\eta^{(l)}$.

Finally, we normalize $\mathbf{A}$ that meets the above two rules and define a fine-grained similarity as follows:

$$\begin{aligned} \mathbf{S} &= \big(\mathbf{A} - \min(\mathbf{A})\big) / \big(\max(\mathbf{A}) - \min(\mathbf{A})\big) \\ &= \frac{a\bar{\mathbf{L}}^\top \bar{\mathbf{L}} + \sum_l \eta^{(l)} \bar{\mathbf{X}}^{(l)\top} \bar{\mathbf{X}}^{(l)} + \sum_l g^{(l)}\eta^{(l)}}{a + \sum_l \eta^{(l)} + \sum_l g^{(l)}\eta^{(l)}}. \end{aligned} \tag{6}$$

Note that $\mathbf{S}$ is calculated on-the-fly by its right side, thus the $O(n^2)$ computational complexity during optimization can be avoided. More importantly, the above fusion strategy is thematically deduced; therefore, there is no conflict between high- and low-level similarities.

3.2.2 Efficient Discrete Optimization. However, Eqn. (1) is NP-hard due to the binary and sparse constraints of hash codes. Moreover, the symmetric inner product of binary codes is not able to reconstruct the subtle differences in the fine-grained similarity. Therefore, we adopt an asymmetric strategy [7, 10, 14] to solve the optimization problem. Specifically, we introduce a real-valued variable $\mathbf{H}$, and replace one $\mathbf{B}$ in the symmetric inner product with $\mathbf{H}$. To minimize the relaxation loss between them, we also introduce a regularization term between them and add an orthogonal constraint on $\mathbf{H}$. Thereafter, Eqn. (1) is transformed into the following one:

$$\begin{aligned} &\min_{\mathbf{B},\mathbf{H}} \|\mathbf{H}^\top\mathbf{B} - r\mathbf{S}\|^2 + \omega\|\mathbf{B} - \mathbf{H}\|^2, \\ &s.t.\ \mathbf{B} \in \{0,1\}^{k\times n},\ \mathbf{B}^\top\mathbf{1}_k = r\mathbf{1}_n,\ \mathbf{H}\mathbf{H}^\top = nr/k\mathbf{I}. \end{aligned} \tag{7}$$

Then the problem can be efficiently solved by an alternative iterative algorithm. The details are described below.

♦ **Update H.** With $\mathbf{B}$ fixed, Eqn. (7) can be rewritten as follows:

$$\begin{aligned} &\max_{\mathbf{H}} \operatorname{tr}\big((r\mathbf{B}\mathbf{S} + \omega\mathbf{B})\mathbf{H}^\top\big), \\ &s.t.\ \mathbf{H}\mathbf{H}^\top = nr/k\mathbf{I}. \end{aligned} \tag{8}$$

We further denote $\mathbf{G} = r\mathbf{B}\mathbf{S} + \omega\mathbf{B}$. Following [30], we perform eigendecomposition of $\mathbf{G}\mathbf{G}^\top$ as follows:

$$\mathbf{G}\mathbf{G}^\top = \begin{bmatrix}\mathbf{V} & \tilde{\mathbf{V}}\end{bmatrix} \begin{bmatrix}\mathbf{\Sigma} & \mathbf{0} \\ \mathbf{0} & \mathbf{0}\end{bmatrix} \begin{bmatrix}\mathbf{V} & \tilde{\mathbf{V}}\end{bmatrix}^\top, \tag{9}$$

where $\mathbf{\Sigma} \in \Re^{r'\times r'}$ is the diagonal matrix of positive eigenvalues, and $\mathbf{V} \in \Re^{r\times r'}$ is the matrix composed of the corresponding eigenvectors. $\tilde{\mathbf{V}}$ is the matrix of remaining $r - r'$ eigenvectors with zero eigenvalue. r' is the rank of $\mathbf{G}\mathbf{G}^\top$. If $r' = r$, $\tilde{\mathbf{V}}$ is empty. Denote $\mathbf{U} = \mathbf{G}^\top\mathbf{V}\mathbf{\Sigma}^{-1/2}$. We then perform the Gram-Schmidt process on $\tilde{\mathbf{V}}$ and a random matrix $\tilde{\mathbf{U}} \in \Re^{n\times(r-r')}$ to generate an orthogonal matrix $\hat{\mathbf{V}} \in \Re^{r\times(r-r')}$ and an orthogonal random matrix $\hat{\mathbf{U}} \in \Re^{n\times(r-r')}$. Thereafter, the optimal solution of Eqn. (8) is,

$$\mathbf{H} = \sqrt{nr/k}\begin{bmatrix}\mathbf{V} & \hat{\mathbf{V}}\end{bmatrix}\begin{bmatrix}\mathbf{U} & \hat{\mathbf{U}}\end{bmatrix}^\top. \tag{10}$$

♦ **Update B.** Fixing $\mathbf{H}$, the problem of Eqn. (7) becomes the following one:

$$\begin{aligned} &\max_{\mathbf{B}} \operatorname{tr}\big((r\mathbf{H}\mathbf{S} + \omega\mathbf{H})\mathbf{B}^\top\big), \\ &s.t.\ \mathbf{B} \in \{0,1\}^{k\times n},\ \mathbf{B}^\top\mathbf{1}_k = r\mathbf{1}_n. \end{aligned} \tag{11}$$

Considering the binary and sparse constrains, the optimal solution to Eqn. (11) is,

$$\mathbf{B} = \operatorname{sign}_r(r\mathbf{H}\mathbf{S} + \omega\mathbf{H}). \tag{12}$$

Here, $\operatorname{sign}_r(\cdot)$ is a function that combines binarization and sparsification, defined as follows:

$$\operatorname{sign}_r(x) = \begin{cases} 1, & \text{if } x \text{ is in top-}r, \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

The $\operatorname{sign}_r(\cdot)$ function is a kind of WTA competition, which activates only the top-r elements to one while all others to zero [25]. In other words, only the top-r elements in each column of $(r\mathbf{H}\mathbf{S} + \omega\mathbf{H})$ are set to one while all others are zero, which can be efficiently computed by a heap sort algorithm.

3.3 Hash Functions Learning

After getting the hash codes $\mathbf{B}$, we further learn the hash functions to map out-of-sample instances into the Hamming space. Generally speaking, this problem can be regarded as a binary classification problem under the supervision of the learnt hash codes [4, 27, 33, 49], and many models can be adopted here, such as support vector machine [21], deep neural networks [53], and so on. Thereinto, the linear model is the most widely-used, which learns hash functions as a linear regression problem as follows:

$$\min_{\mathbf{W}^{(l)}} \|\mathbf{B} - \mathbf{W}^{(l)}\mathbf{X}^{(l)}\|^2 + \lambda\|\mathbf{W}^{(l)}\|^2, \tag{14}$$

Algorithm 1 High-dimensional Sparse Cross-modal Hashing

Input: Training data $\mathbf{X}^{(l)}$ ($l \in \{1, 2, ..., m\}$), label matrix $\mathbf{L}$, dense code length r, activity ratio τ, maximum iteration number t, parameters ω and λ.

Output: Sparse hash codes $\mathbf{B}$, hash function $H_l(\mathbf{x}^{(l)}_{query})$.

% Step-1: Hash Codes Learning.

1. Initialize $\mathbf{B}$, $\mathbf{H}$ randomly with a standard normal distribution;

repeat

2. Update $\mathbf{H}$ using Eqn. (10);

3. Update $\mathbf{B}$ using Eqn. (12);

until convergent or maximum iterations

% Step-2: Hash Functions Learning.

4. Compute the mapping matrix $\mathbf{W}^{(l)}$ using Eqn. (15);

return Sparse hash codes $\mathbf{B}$, hash function $H_l(\mathbf{x}^{(l)}_{query})$.

where λ is a penalty parameter to avoid overfitting. Formally, Eqn. (14) has a closed-form solution by setting its derivative with respect to $\mathbf{W}^{(l)}$ to zero. Afterwards, we have the optimal solution as follows:

$$\mathbf{W}^{(l)} = \mathbf{B}\mathbf{X}^{(l)\top}(\mathbf{X}^{(l)}\mathbf{X}^{(l)\top} + \lambda\mathbf{I})^{-1}. \tag{15}$$

Then, given a query of the l-th modality, its hash function is,

$$H_l(\mathbf{x}^{(l)}_{query}) = \text{sign}_r(\mathbf{W}^{(l)}\mathbf{x}^{(l)}_{query}). \tag{16}$$

To sum up, the whole training process including hash codes learning and hash functions learning is described in Algorithm 1.

3.4 Convergence Proof

In this section, we give a theoretical analysis on the convergence of HSCH. Firstly, for the hash codes learning step, we design an alternative iterative algorithm. Its convergence can be proved as follows. Let $\mathcal{L}(\mathbf{H}, \mathbf{B})$ denote the objective function in Eqn. (7). Due to each variable has a closed-form solution in their corresponding subproblems, we have $\mathcal{L}(\mathbf{H}^{t'+1}, \mathbf{B}^{t'+1}) \leq \mathcal{L}(\mathbf{H}^{t'+1}, \mathbf{B}^{t'}) \leq \mathcal{L}(\mathbf{H}^{t'}, \mathbf{B}^{t'})$, where t' is the iterative round. The objective function is a summation of positive norms and its value is monotonously decreasing. According to the theorem in [38], the optimization algorithm could produce a convergent solution. Secondly, for the hash functions learning step, it has a closed-form solution by setting the derivative to zero. In brief, the learning of HSCH is theoretically convergent.

3.5 Complexity Analysis

In this section, we analyse the computation and space cost of HSCH. During the training phase, the computational complexity of HSCH includes $O(ckn + \sum_l d_l kn + kn + k^2 n + k^3 + r^2 n)$ for updating $\mathbf{H}$, $O(ckn + \sum_l d_l kn + kn + nk\log_2 r)$ for updating $\mathbf{B}$, and $O(d_l^2 + d_l + d_l^2 k + d_l kn + d_l^2 n)$ for computing $\mathbf{W}^{(l)}$, respectively, where k is the dimensionality of sparse codes, r is the code length, d_l is the feature dimensionality of l-th modality, c is the class number, and n is the training scale. Overall, the entire computational complexity is $O\big(t(ck + \sum_l d_l k + k + k^2 + r^2 + ck + k\log_2 r)n + \sum_l (d_l k + d_l^2)n + tk^3 + \sum_l (d_l^2 + d_l + d_l^2 k)\big)$ for training HSCH, where t is the maximum iterations, and $l \in \{1, ..., m\}$. Generally speaking, $r, k, d_l, c, t, m \ll n$, the computational complexity of HSCH is linear to the size of the training set, i.e., $O(n)$, which is scalable to large-scale datasets. From the aspect of space cost, the fine-grained similarity defined in Eqn. (6) is calculated on-the-fly. Besides, the high-level similarity $\mathbf{A}^{(s)}$, the low-level similarity $\mathbf{A}^{(f)}$, and the fused affinity $\mathbf{A}$ are defined for ease of representation. Apart from them, the size of all other variables is linear or irrelevant to n. As a consequence, the space cost of HSCH is also linear to the size of the training set.

During the retrieval procedure, for a given query of the l-th modality, the computational complexity of HSCH includes $O(kd_l + k\log_2 r)$ for generating a sparse code, and $O(rN)$ for calculating Hamming distance, where N is the size of retrieval database. As a comparison, for a general dense hashing method, its computational complexity usually contains $O(rd_l + r)$ for generating a dense hash code and $O(rN)$ for calculating Hamming distance. Apparently, the computational cost of HSCH used to calculate Hamming distance is the same as that of dense ones. Note that for a large-scale database, $k \ll N$; thereby the cost of generating binary codes is negligible compared to calculating Hamming distance. In addition, the space cost of HSCH is $O(r\log_2 k)$ for one high-dimensional sparse binary code, while $O(r)$ for dense hashing. Although HSCH has a slightly higher space cost than dense ones, large-scale storage arrays are relatively cheaper than accuracy and query speed.

4 EXPERIMENTS

4.1 Experimental Settings

4.1.1 Datasets. To evaluate the performance of HSCH, we conducted extensive experiments on three widely-used datasets, i.e., MIRFlickr-25K [17], IAPR TC-12 [11], and NUS-WIDE [6]. Their experimental settings are described below.

MIRFlickr-25K: It includes 25,000 image-text pairs. Following [20], we removed those instances that have no labels. As a result, 20,015 instances are used. Each image-text pair is multi-labeled with 24 classes, and is represented by a 512-dimensional GIST feature vector for image modality and a 1,386-dimensional bag-of-words vector for text modality. We randomly selected 2,000 samples as the query set and the remaining as the retrieval and training set.

IAPR TC-12: It contains 20,000 samples of image and text modalities, with each sample belonging to at least one of 255 categories. Each image is represented by a 512-dimensional GIST feature vector, and each text is represented by a 2,912-dimensional bag-of-words vector. We randomly selected 2,000 image-text pairs as the query set and the rest as the retrieval and training set.

NUS-WIDE: It is a real-world dataset, which consists of 269,468 images associated with text captions. In our experiments, only 186,577 instances belonging to the top-10 most frequent labels are utilized. Each image is represented by a 500-dimensional SIFT feature vector, while each associated text caption is a 1,000-dimensional binary tagging vector. We randomly selected 2,000 samples as the query set and the left as the retrieval and training set.

4.1.2 Evaluation Metrics. We conducted two types of cross-modal retrieval tasks: 1) Image-to-Text ($I \rightarrow T$), i.e., using an image as a query to retrieve similar texts; 2) Text-to-Image ($T \rightarrow I$), i.e., using a text description as a query to search for similar images. The retrieval performance of HSCH is evaluated by several widely-used evaluation metrics, such as mean average precision (MAP),

Table 1: The MAP results of HSCH and baselines on three benchmarks.

Task	Method	MIRFlickr-25K					IAPR TC-12					NUS-WIDE				
		2-bit	4-bit	8-bit	16-bit	32-bit	2-bit	4-bit	8-bit	16-bit	32-bit	2-bit	4-bit	8-bit	16-bit	32-bit
$I \rightarrow T$	DCH	0.6682	0.6994	0.6974	0.7154	0.7255	0.3561	0.3928	0.4448	0.4666	0.4817	0.4498	0.5695	0.5913	0.6421	0.6037
	SCRATCH	0.6632	0.6789	0.6978	0.7055	0.7149	0.3695	0.4066	0.4334	0.4499	0.4730	0.5006	0.5979	0.6122	0.6424	0.6462
	LCMFH	0.6494	0.6454	0.6834	0.6879	0.6981	0.3768	0.4090	0.4174	0.4259	0.4434	0.5495	0.5915	0.6133	0.6259	0.6548
	DLFH	0.5569	0.6657	0.6807	0.7112	0.7206	0.3032	0.3522	0.3963	0.3955	0.4392	0.3403	0.5304	0.5820	0.6353	0.6554
	HSCH	**0.7242**	**0.7467**	**0.7586**	**0.7676**	**0.7766**	**0.4564**	**0.5023**	**0.5231**	**0.5416**	**0.5452**	**0.6501**	**0.6624**	**0.6662**	**0.6802**	**0.6868**
$T \rightarrow I$	DCH	0.7010	0.7514	0.7505	0.7765	0.7762	0.3910	0.4398	0.5043	0.5503	0.5746	0.5145	0.6808	0.7456	0.7566	0.7408
	SCRATCH	0.6837	0.7210	0.7549	0.7703	0.7829	0.3805	0.4401	0.4917	0.5370	0.5792	0.5120	0.6831	0.7347	0.7684	0.7809
	LCMFH	0.6819	0.6848	0.7409	0.7460	0.7648	0.3873	0.4311	0.4553	0.4885	0.5140	0.6128	0.6974	0.7578	0.7684	0.7956
	DLFH	0.5571	0.7195	0.7423	0.7929	0.8134	0.3032	0.3481	0.4006	0.4410	0.4963	0.3402	0.6111	0.7377	0.7641	0.8104
	HSCH	**0.7805**	**0.8175**	**0.8342**	**0.8441**	**0.8544**	**0.5383**	**0.6011**	**0.6407**	**0.6620**	**0.6719**	**0.7522**	**0.7752**	**0.7884**	**0.8004**	**0.8130**

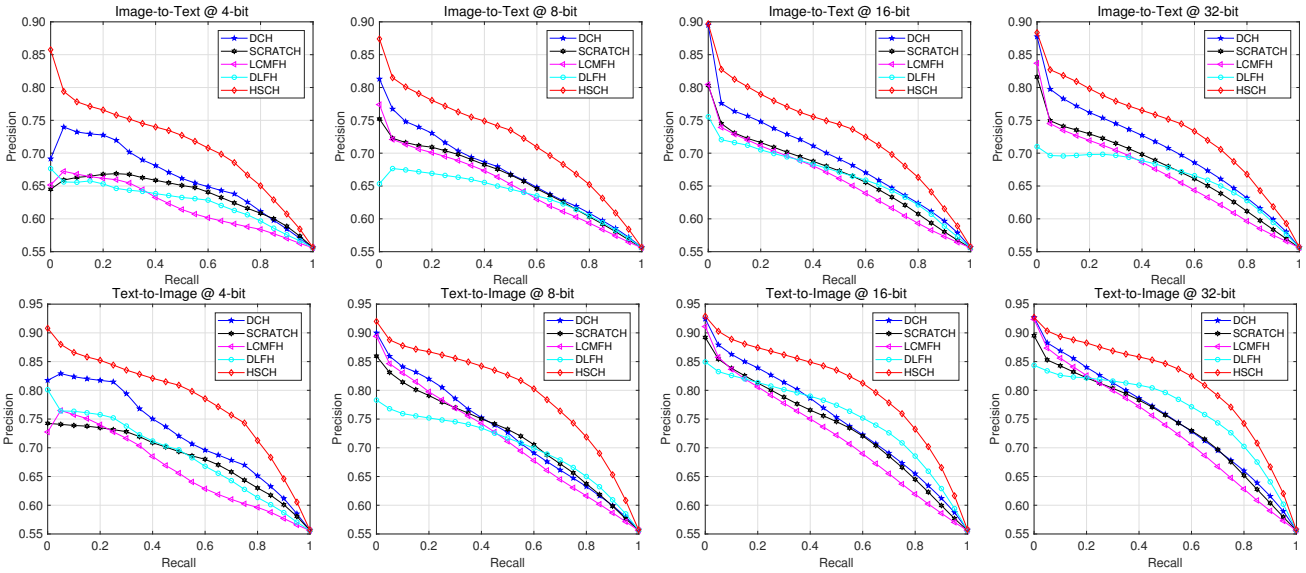

Figure 1: The precision-recall curves of all methods on MIRFlickr-25K.

precision-recall curve, top-N precision curve, and time cost. The retrieval list is ordered according to the Hamming distance between the hash codes of query and database. Two points are regarded as similar if they share at least one common label. We calculate average precision (AP) and then average its values as MAP. The precision-recall curve indicates the relationship between precision and recall. The top-N precision curve reflects the change of precision with respect to the number of top retrieved samples. Generally speaking, for the above three metrics, the larger the value, the better the accuracy. In addition, the time cost includes training time and retrieval time. The smaller the time, the better the efficiency.

We reported these results on various code length r. It is worth noting that the code length r here indicates the dimensionality of dense codes for dense hash methods and the number of nonzero elements for HSCH. In fact, the code dimensionality of HSCH is $k = r/\tau$, where τ is the activity ratio.

4.1.3 Baselines and Implementation Details. In our experiments, four supervised cross-modal hashing methods are utilized as baselines, i.e., DCH [54], SCRATCH [3], LCMFH [46], and DLFH [20]. They all belong to dense hashing, which uses r-bit dense hash codes. Notably, there is no sparse cross-modal hashing method so far. Thus, we only compare HSCH with dense baselines. For DCH, SCRATCH, and DLFH, they are implemented based on the source codes kindly provided by their authors. For LCMFH, we carefully implemented it with the suggested parameters in the paper. For our HSCH, the parameters are set experimentally, i.e., $\omega = 10$, $\lambda = 0.01$. The activity ratio $\tau = r/k$ is set to 5%, and the maximum iterative number t is set to 5. All our experiments are conducted on a Linux workstation with Intel XEON E5-2650 2.20GHz CPU, 128GB RAM.

4.2 Results and Discussion

4.2.1 Accuracy Comparison. The MAP results of all methods on three benchmark datasets are shown in Table 1, including the Image-to-Text and Text-to-Image tasks. The code length varies from 2 to 32. The best results are shown in boldface. Furthermore, we plotted the precision-recall and top-N precision curves on MIRFlickr-25K with the case of 4, 8, 16, and 32 bits in Figure 1 and 2, respectively. From these results, we have the following observations:

- In light of the MAP results, HSCH obtains the best performance, significantly outperforming all baselines, especially at short code length. This phenomenon indicates the powerful capability of high-dimensional sparse codes in HSCH.

Further considering the query speed, HSCH is a good choice when short code lengths are required. It is worth remembering that HSCH consumes a space cost of $r \log_2 k$ instead of r per entry. Even in scenarios where space cost is limited, HSCH at 2 bits, with a space cost of $2 \times \log_2(2/0.05) \approx 10.6$, is competitive with other methods at 16 or 32 bits, yet shorter code length costs cheaper Hamming distance computation.
- With the code length increasing, the results of most methods have some improvements, confirming the fact that more bits could carry more supervised information. Nevertheless, the performance of HSCH saturates at about 16 bits, with only a small improvement from 16 to 32 bits. This also demonstrates the effectiveness of HSCH at short code length.
- In Figure 1 and 2, HSCH substantially outperforms all baselines in large gaps, which are consistent with the trends of MAP results in Table 1.

4.2.2 Efficiency Comparison. In addition, we further reported the training time and retrieval time of HSCH and all baselines on three datasets in Table 2. The training time includes both the time of learning hash codes of training samples and that of learning hash functions. The retrieval time includes generating hash codes of a query and calculating Hamming distance between the hash codes of the query and samples in the retrieval database. From these results, we have the following observations:

- Concerning the training time, HSCH achieves comparable or better results than several baselines. It is worth noting that the complexity of all methods is linear to the size of training set. Compared with them, HSCH still achieves practicable time cost, not to mention its code dimensionality is much higher than those dense baselines. Moreover, despite all methods having linear complexity, their training time is very diverse. One reason is that their optimization algorithms are different. For example, DCH applies the DCC algorithm, which is much more time-consuming; LCMFH adopts a relaxation strategy, which is faster than discrete optimization. Furthermore, another reason is that these methods have different sensitivity to code length r, feature dimensionality d_l, class number c, iterations t, and so on.
- With respect to the retrieval time, HSCH reports similar results as those dense baselines, confirming the complexity analysis in Section 3.5. In addition, since NUS-WIDE is nearly ten times larger than MIRFlickr-25K and IAPR TC-12, correspondingly, the time costs of all methods on NUS-WIDE are nearly ten times greater than those on the other two datasets, confirming the fact that the computational complexity of retrieval is linear to the size of retrieval database.
- Jointly considering the training and retrieval time, we can conclude that HSCH is efficient and scalable to large-scale datasets.

4.2.3 Ablation Study. To gain a deep insight into HSCH, we further conducted ablation experiments on three benchmarks, including three variations of HSCH, i.e., DenseCH-S, DenseCH, and HSCH-S. Thereinto, 'DenseCH-S' means using dense hash codes and binary similarity matrix; 'DenseCH' means using dense hash codes and the fine-grained similarity matrix; 'HSCH-S' means using high-dimensional sparse codes and binary similarity matrix. Due to the binary similarity matrix will incur $O(n^2)$ problem, we randomly sampled 5,000 instances from retrieval set as the training set of DenseCH-S and HSCH-S. The MAP results of HSCH and its variations are shown in Table 3 with the code length varying from 2 to 32. From this table, we have the following observations:

- DenseCH has a great performance gain over DenseCH-S. One possible reason is that DenseCH avoids the problem of $O(n^2)$ complexity, so all available data is utilized to train the model. Another reason is that the fine-grained similarity in DenseCH carries more information than the binary one in DenseCH-S.
- Compared with DenseCH, HSCH has been greatly improved, owing to the powerful representation capability of high-dimensional sparse codes. When the code length is short, the performance gain of HSCH over DenseCH is much larger than that when the code length is long. This phenomenon further verifies that HSCH has significantly large advantages at short code length. With the code length increasing, HSCH and DenseCH may tend to be the same. However, when we pursue the same accuracy, HSCH can greatly improve the query speed, which is a good choice for those applications that require high efficiency. Besides, we believe that for hashing, an approximate nearest neighbor search strategy, search efficiency is more important than extreme accuracy.
- Between DenseCH-S and HSCH-S, the improvements of HSCH-S are not obvious. One possible reason is that they are both trained on only 5,000 samples, so the power of sparse codes is limited. This phenomenon in turn shows that HSCH is more suitable for large-scale datasets.
- Jointly considering Table 1 and 3, we can see that DenseCH is still superior to all dense baselines, further validating the effectiveness of the well-designed similarity in HSCH. In conclusion, the combination of high-dimensional sparse codes and the fine-grained similarity complement each other, greatly improving the performance of HSCH.

4.2.4 Parameter Sensitivity Analysis. We also conducted experiments on MIRFlickr-25K to analyse the parameter sensitivity of HSCH. There are three parameters in HSCH, i.e., τ, ω, and λ. Thereinto, λ is used for regularization, which is set empirically; thus, to save space, we only show the results about ω and τ by varying the value of one parameter while fixing another. The MAP results of HSCH on MIRFlickr-25K in the cases of Image-to-Text and Text-to-Image tasks are plotted in Figure 3. The hash code length r is 8 bits. We can observe that: 1) The influence of ω is not significant, when its value is not very large. The reason is that the term of ω is used to control relaxation loss, setting as a large value will severely bias the optimization. HSCH obtains the best results when ω equals to 10. For simplicity, we set the same value for all benchmark datasets. 2) τ also affects the results. In detail, the smaller the activity ratio, i.e., the higher the code dimensionality k, the higher the accuracy. Besides, the performance improvement from 0.5% to 0.25% is small, indicating the performance would saturate when the activity ratio is too small. Whereas, higher code dimensionality will result in

Table 2: The time costs (second) of HSCH and baselines on three benchmarks.

Process	Method	MIRFlickr-25K					IAPR TC-12					NUS-WIDE				
		2-bit	4-bit	8-bit	16-bit	32-bit	2-bit	4-bit	8-bit	16-bit	32-bit	2-bit	4-bit	8-bit	16-bit	32-bit
Training	DCH	3.3685	1.3809	1.5331	1.7195	3.5618	9.6790	1.5654	2.5072	2.2983	9.7310	28.066	15.173	13.884	17.246	44.737
	SCRATCH	2.6399	2.5172	2.6527	2.6727	2.9199	3.3180	2.8321	2.7940	2.8840	3.4092	24.299	24.503	24.542	26.124	27.600
	LCMFH	2.0045	1.9747	1.9066	2.0004	2.1754	3.4171	3.6562	3.6183	3.5420	2.9592	18.762	19.530	19.832	20.735	20.469
	DLFH	0.4761	0.5933	0.9674	2.4638	6.9969	0.5505	0.6435	1.0178	2.3046	6.5387	3.9603	4.6633	7.5136	27.964	107.25
	HSCH	1.0612	1.0583	1.7785	3.2905	7.1237	0.9515	1.1538	1.7159	3.3362	6.5849	8.9938	10.387	17.487	31.212	58.118
Retrieval	DCH	0.0390	0.0493	0.0506	0.0495	0.0542	0.0394	0.0484	0.0512	0.0496	0.0428	0.3544	0.3677	0.3905	0.3750	0.4021
	SCRATCH	0.0391	0.0456	0.0502	0.0509	0.0512	0.0484	0.0489	0.0508	0.0491	0.0518	0.3360	0.3427	0.3610	0.3958	0.3999
	LCMFH	0.0523	0.0501	0.0516	0.0532	0.0583	0.0503	0.0504	0.0523	0.0542	0.0513	0.3565	0.3931	0.3675	0.4578	0.4954
	DLFH	0.0492	0.0491	0.0464	0.0493	0.0564	0.0512	0.0512	0.0576	0.0573	0.0571	0.3543	0.3638	0.3758	0.4111	0.4812
	HSCH	0.0440	0.0449	0.0475	0.0496	0.0539	0.0454	0.0449	0.0477	0.0468	0.0491	0.3052	0.3113	0.3207	0.3373	0.4277

Table 3: The ablation results of HSCH on three benchmarks.

Task	Method	MIRFlickr-25K					IAPR TC-12					NUS-WIDE				
		2-bit	4-bit	8-bit	16-bit	32-bit	2-bit	4-bit	8-bit	16-bit	32-bit	2-bit	4-bit	8-bit	16-bit	32-bit
$I \rightarrow T$	DenseCH-S	0.6258	0.6240	0.6525	0.6563	0.6574	0.3279	0.3354	0.3440	0.3585	0.3680	0.4506	0.4652	0.4727	0.4949	0.5061
	DenseCH	0.6569	0.6904	0.7170	0.7446	0.7498	0.3381	0.3972	0.4153	0.4668	0.4976	0.4492	0.6077	0.6454	0.6524	0.6569
	HSCH-S	0.6136	0.6339	0.6466	0.6535	0.6604	0.3592	0.3621	0.3776	0.3727	0.3784	0.4785	0.5062	0.5272	0.5345	0.5324
	HSCH	**0.7242**	**0.7467**	**0.7586**	**0.7676**	**0.7766**	**0.4564**	**0.5023**	**0.5231**	**0.5416**	**0.5452**	**0.6501**	**0.6624**	**0.6662**	**0.6802**	**0.6868**
$T \rightarrow I$	DenseCH-S	0.6213	0.6387	0.6525	0.6525	0.6565	0.3356	0.3521	0.3655	0.3892	0.3984	0.4386	0.4549	0.4665	0.4768	0.4894
	DenseCH	0.6990	0.7370	0.7917	0.8190	0.8318	0.3606	0.4278	0.4805	0.5563	0.5996	0.5136	0.7024	0.7625	0.7784	0.7865
	HSCH-S	0.6158	0.6385	0.6485	0.6558	0.6592	0.3747	0.3836	0.3983	0.3998	0.4077	0.4535	0.4748	0.4940	0.5006	0.5056
	HSCH	**0.7805**	**0.8175**	**0.8342**	**0.8441**	**0.8544**	**0.5383**	**0.6011**	**0.6407**	**0.6620**	**0.6719**	**0.7522**	**0.7752**	**0.7884**	**0.8004**	**0.8130**

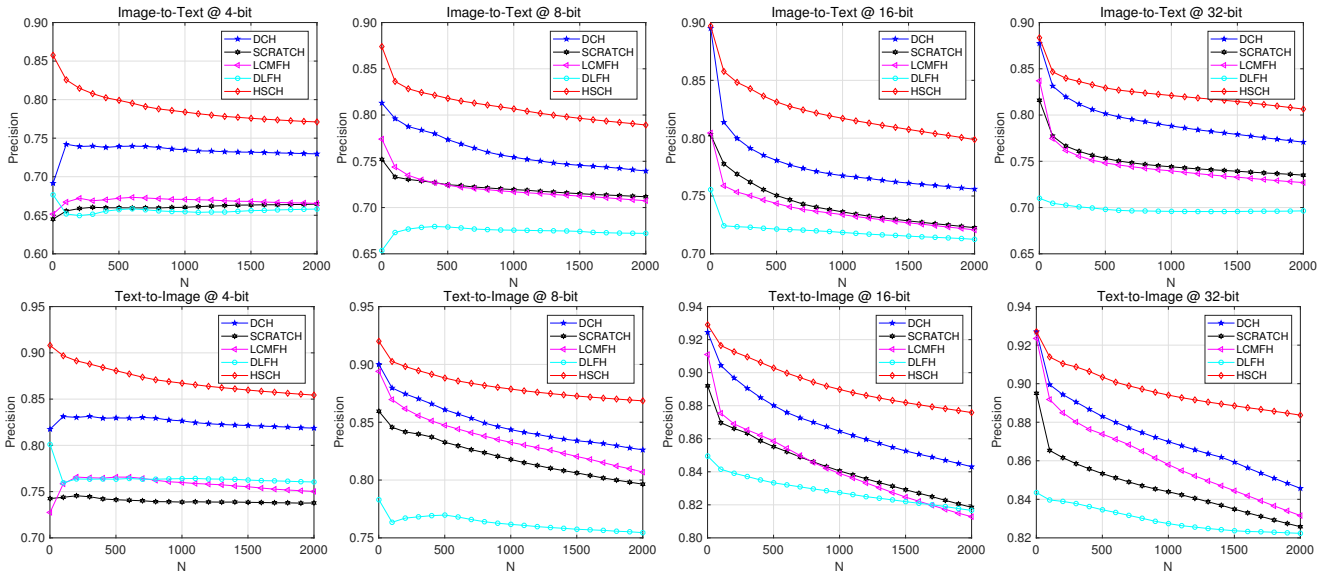

Figure 2: The top-N precision curves of all methods on MIRFlickr-25K.

higher training cost. To balance the accuracy and training efficiency, we experimentally choose 5% as the value of activity ratio τ.

4.2.5 Convergence Analysis. In Section 3.4, we theoretically analyse the convergence of HSCH. To have a deep insight, we further conducted experiments to show the convergence of the efficient discrete optimization algorithm described in Section 3.2.2. Specifically, we plotted the normalized objective values with respect to the iterations on MIRFlickr-25K and NUS-WIDE with the cases of 2, 4, 8, 16, and 32 bits in Figure 4. From this figure, we can see that the values drop sharply after one iteration, and the algorithm always converges within 5 iterations, confirming its convergence and efficiency.

4.2.6 Comparison with Deep Hashing. We also compared HSCH with several state-of-the-art deep cross-modal hashing methods, i.e., DCMH [19], SSAH [24], EGDH [42], MLCAH [36], and DADH [1]. Particularly, the features of image modality used by HSCH$_{cnn}$ are replaced with deep features extracted from CNN-F model [53]

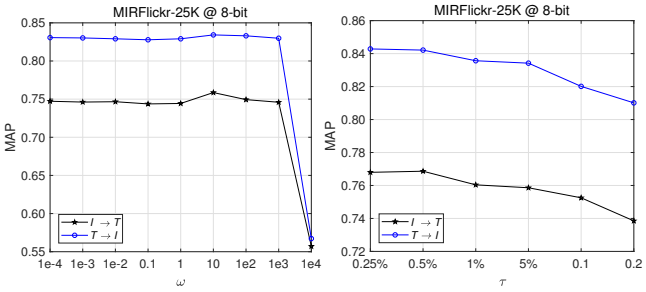

Figure 3: Parameter analysis of HSCH on MIRFlickr-25K.

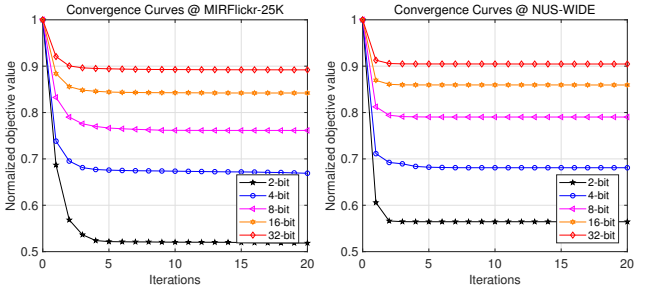

Figure 4: Convergence analysis of the discrete optimization algorithm on MIRFlickr-25K and NUS-WIDE.

Table 4: The MAP results of HSCH with CNN features and deep baselines on MIRFlickr-25K.

Task	Method	16-bit	32-bit	64-bit
$I \rightarrow T$	DCMH[19]	0.7410	0.7465	0.7485
	SSAH[24]	0.7820	0.7900	0.8000
	EGDH[42]	0.7569	0.7729	0.7959
	MLCAH[36]	0.7960	0.8080	0.8150
	DADH[1]	0.8020	0.8072	0.8179
	HSCH_{cnn}	**0.8646**	**0.8735**	**0.8760**
$T \rightarrow I$	DCMH[19]	0.7827	0.7900	0.7932
	SSAH[24]	0.7910	0.7950	0.8030
	EGDH[42]	0.7787	0.7939	0.7985
	MLCAH[36]	0.7940	0.8050	0.8050
	DADH[1]	0.7920	0.7959	0.8064
	HSCH_{cnn}	**0.8475**	**0.8558**	**0.8587**

pre-trained on ImageNet dataset [39]. The MAP results with various code lengths are summarized in Table 4. The results of all baselines are those reported in the original papers. From this table, we can see that HSCH_{cnn} is superior to all deep baselines by a large gap. Although HSCH_{cnn} is not an end-to-end deep model, it still achieves better performance, demonstrating the effectiveness of fine-grained similarity and high-dimensional sparse codes.

5 CONCLUSION

In this paper, we present a novel High-dimensional Sparse Cross-modal Hashing method, termed as HSCH, which mainly focuses on how to fully exploit the information of data to seek fine-grained retrieval results and how to improve the representation capability of hash codes. Specifically, we first design a fine-grained similarity with two innovative rules, taking both high-level semantic similarity and low-level feature similarity into consideration. Then we explore the power of high-dimensional sparse codes to embed the fine-grained similarity into the to-be-learnt hash codes. In light of this, the learnt hash codes are able to carry more information, accordingly leading to more fine-grained retrieval results. Additionally, HSCH discretely solves the binary and sparse constraints, avoiding the quantization error problem. More importantly, even the code dimensionality of HSCH is higher than dense ones, its training complexity is linear to the size of training set and its retrieval complexity is the same as that of dense hashing, making it scalable to large-scale datasets. Extensive experiments on three benchmarks demonstrate the state-of-the-art performance of HSCH. Inspired by the results in Section 4.2.6, we plan to develop an end-to-end deep model in future work.

ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China under Grant 61872428, in part by Shandong Provincial Key Research and Development Program under Grant 2019JZZY010127, in part by Natural Science Foundation of Shandong Province under Grant ZR2019ZD06, in part by the Major Program of the National Natural Science Foundation of China under Grant 61991411, in part by Natural Science Foundation of Shandong Province under Grant ZR2020QF036, and in part by the Fundamental Research Funds of Shandong University under Grant 2019GN075.

REFERENCES

[1] Cong Bai, Chao Zeng, Qing Ma, Jinglin Zhang, and Shengyong Chen. 2020. Deep adversarial discrete hashing for cross-modal retrieval. In *Proc. ACM Int. Conf. Multimedia Retr.* 525–531.
[2] Moses S. Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *Proc. ACM Symposium on Theory of Compu.* 380–388.
[3] Zhen Duo Chen, Chuan Xiang Li, Xin Luo, Liqiang Nie, Wei Zhang, and Xin Shun Xu. 2020. SCRATCH: A scalable discrete matrix factorization hashing framework for cross-modal retrieval. *IEEE Trans. Circuits Syst. Video Technol.* 30, 7 (2020), 2262–2275.
[4] Zhen Duo Chen, Yongxin Wang, Hui Qiong Li, Xin Luo, Liqiang Nie, and Xin Shun Xu. 2019. A two-step cross-modal hashing by exploiting label correlations and preserving similarity in both steps. In *Proc. ACM Multimedia Conf.* 1694–1702.
[5] Miaomiao Cheng, Liping Jing, and Michael K. Ng. 2020. Robust unsupervised cross-modal hashing for multimedia retrieval. *ACM Trans. Inf. Syst.* 38, 3 (2020), 1–25.
[6] Tat Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. 2009. NUS-WIDE: A real-world web image database from National University of Singapore. In *Proc. ACM Int. Conf. Image Video Retr.* 48.
[7] Cheng Da, Shibiao Xu, Kun Ding, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. 2017. Asymmetric multi-valued hashing. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.* 736–744.
[8] Sanjoy Dasgupta, Charles F. Stevens, and Saket Navlakha. 2017. A neural algorithm for a fundamental computing problem. *Science* 358, 6364 (2017), 793–796.
[9] Guiguang Ding, Yuchen Guo, and Jile Zhou. 2014. Collective matrix factorization hashing for multimodal data. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.* 2075–2082.
[10] Wei Dong, Moses Charikar, and Kai Li. 2008. Asymmetric distance estimation with sketches for similarity search in high-dimensional spaces. In *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.* 123–130.
[11] Hugo Jair Escalante, Carlos A. Hernandez, Jesus A. Gonzalez, Aurelio Lopez-lopez, Manuel Montes-y-Gómez, Eduardo F. Morales, Luis Enrique Sucar, Luis Villasenor, and Michael Grubinger. 2010. The segmented and annotated IAPR TC-12 benchmark. *Comput. Vision Image Understanding* 114, 4 (2010), 419–428.
[12] Jianlong Fu, Heliang Zheng, and Tao Mei. 2017. Look closer to see better: recurrent attention convolutional neural network for fine-grained image recognition. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.* 4476–4484.
[13] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. 2013. Iterative quantization: A procrustean approach to learning binary codes for

large-scale image retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.* 35, 12 (2013), 2916–2929.
[14] Albert Gordo, Florent Perronnin, Yunchao Gong, and Svetlana Lazebnik. 2014. Asymmetric distances for binary embeddings. *IEEE Trans. Pattern Anal. Mach. Intell.* 36, 1 (2014), 33–47.
[15] Hengtong Hu, Lingxi Xie, Richang Hong, and Qi Tian. 2020. Creating something from nothing: unsupervised knowledge distillation for cross-modal hashing. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.* 3123–3132.
[16] Mengqiu Hu, Yang Yang, Fumin Shen, Ning Xie, Richang Hong, and Heng Tao Shen. 2019. Collective reconstructive embeddings for cross-modal hashing. *IEEE Trans. Image Process.* 28, 6 (2019), 2770–2784.
[17] Mark J. Huiskes and Michael S. Lew. 2008. The MIR flickr retrieval evaluation. In *Proc. ACM Int. Conf. on Multimedia Inf. Retr.* 39–43.
[18] Rongrong Ji, Hong Liu, Liujuan Cao, Di Liu, Yongjian Wu, and Feiyue Huang. 2017. Toward optimal manifold hashing via discrete locally linear embedding. *IEEE Trans. Image Process.* 26, 11 (2017), 5411–5420.
[19] Qing Yuan Jiang and Wu Jun Li. 2017. Deep cross-modal hashing. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.* 3270–3278.
[20] Qing Yuan Jiang and Wu Jun Li. 2019. Discrete latent factor model for cross-modal hashing. *IEEE Trans. Image Process.* 28, 7 (2019), 3490–3501.
[21] Wang Cheng Kang, Wu Jun Li, and Zhi Hua Zhou. 2016. Column sampling based discrete supervised hashing. In *Proc. AAAI Conf. Artif. Intell.* 1230–1236.
[22] Dmitry Krotov and John J. Hopfield. 2019. Unsupervised learning by competing hidden units. *Proc. Natl. Acad. Sci. USA* 116, 16 (2019), 7723–7731.
[23] Zhihui Lai, Yudong Chen, Jian Wu, Wai Keung Wong, and Fumin Shen. 2018. Jointly sparse hashing for image retrieval. *IEEE Trans. Image Process.* 27, 12 (2018), 6147–6158.
[24] Chao Li, Cheng Deng, Ning Li, Wei Liu, Xinbo Gao, and Dacheng Tao. 2018. Self-supervised adversarial hashing networks for cross-modal retrieval. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.* 4242–4251.
[25] Wenye Li, Jingwei Mao, Yin Zhang, and Shuguang Cui. 2018. Fast similarity search via optimal sparse lifting. In *Proc. Neural Inf. Process. Syst.* 176–184.
[26] Andrew C. Lin, Alexei M. Bygrave, Alix De Calignon, Tzumin Lee, and Gero MiesenböCk. 2014. Sparse, decorrelated odor coding in the mushroom body enhances learned odor discrimination. *Nat. Neurosci.* 17, 4 (2014), 1097–6256.
[27] Guosheng Lin, Chunhua Shen, David Suter, and Anton van den Hengel. 2013. A general two-step approach to learning-based hashing. In *Proc. IEEE Int. Conf. Comput. Vis.* 2552–2559.
[28] Zijia Lin, Guiguang Ding, Mingqing Hu, and Jianmin Wang. 2015. Semantics-preserving hashing for cross-view retrieval. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.* 3864–3872.
[29] Hong Liu, Rongrong Ji, Yongjian Wu, Feiyue Huang, and Baochang Zhang. 2017. Cross-modality binary code learning via fusion similarity hashing. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.* 7380–7388.
[30] Wei Liu, Cun Mu, Sanjiv Kumar, and Shih Fu Chang. 2014. Discrete graph hashing. In *Proc. Neural Inf. Process. Syst.* 3419–3427.
[31] Wei Liu, Jun Wang, Rongrong Ji, and Yu Gang Jiang. 2012. Supervised hashing with kernels. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.* 2074–2081.
[32] Mingsheng Long, Yue Cao, Jianmin Wang, and Philip S. Yu. 2016. Composite correlation quantization for efficient multimodal retrieval. In *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.* 579–588.
[33] Xin Luo, Liqiang Nie, Xiangnan He, Ye Wu, Zhen Duo Chen, and Xin Shun Xu. 2018. Fast scalable supervised hashing. In *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.* 735–744.
[34] Xin Luo, Peng Fei Zhang, Zi Huang, Liqiang Nie, and Xin Shun Xu. 2019. Discrete hashing with multiple supervision. *IEEE Trans. Image Process.* 28, 6 (2019), 2962–2975.
[35] Changyi Ma, Chonglin Gu, Wenye Li, and Shuguang Cui. 2020. Large-scale image retrieval with sparse binary projections. In *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.* 1817–1820.
[36] Xinhong Ma, Tianzhu Zhang, and Changsheng Xu. 2020. Multi-level correlation adversarial hashing for cross-modal retrieval. *IEEE Trans. Multimedia* 22, 12 (2020), 3101–3114.
[37] Devraj Mandal, Kunal N. Chaudhury, and Soma Biswas. 2019. Generalized semantic preserving hashing for cross-modal retrieval. *IEEE Trans. Image Process.* 28, 1 (2019), 102–112.
[38] Walter Rudin. 1976. *Principles of mathematical analysis*. Vol. 3. McGraw-hill New York.
[39] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei Fei Li. 2015. ImageNet large scale visual recognition challenge. *Int. J. Comput. Vision* 115, 3 (2015), 211–252.
[40] Chaitanya K. Ryali, John J. Hopfield, Leopold Grinberg, and Dmitry Krotov. 2020. Bio-Inspired hashing for unsupervised similarity search. In *Proc. Int. Conf. Mach. Learn.* 8295–8306.
[41] Fumin Shen, Chunhua Shen, Wei Liu, and Heng Tao Shen. 2015. Supervised discrete hashing. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.* 37–45.
[42] Yufeng Shi, Xinge You, Feng Zheng, Shuo Wang, and Qinmu Peng. 2019. Equally-guided discriminative hashing for cross-modal retrieval. In *Proc. Int. Joint Conf. Artif. Intell.* 4767–4773.
[43] Jingkuan Song, Yang Yang, Yi Yang, Zi Huang, and Heng Tao Shen. 2013. Inter-media hashing for large-scale retrieval from heterogeneous data sources. In *Proc. ACM SIGMOD Int. Conf. Manag.* 785–796.
[44] Charles F. Stevens. 2016. A statistical property of fly odor responses is conserved across odors. In *Proc. Natl. Acad. Sci.* 6737—-6742.
[45] Di Wang, Xinbo Gao, Xiumei Wang, and Lihuo He. 2015. Semantic topic multimodal hashing for cross-media retrieval. In *Proc. Int. Joint Conf. Artif. Intell.* 3890–3896.
[46] Di Wang, Xinbo Gao, Xiumei Wang, and Lihuo He. 2019. Label consistent matrix factorization hashing for large-scale cross-modal similarity search. *IEEE Trans. Pattern Anal. Mach. Intell.* 41, 10 (2019), 2466–2479.
[47] Di Wang, Quan Wang, and Xinbo Gao. 2018. Robust and flexible discrete hashing for cross-modal similarity search. *IEEE Trans. Circuits Syst. Video Technol.* 28, 10 (2018), 2703–2715.
[48] Lu Wang, Chao Ma, Enmei Tu, Jie Yang, and Nikola Kasabov. 2018. Discrete sparse hashing for cross-modal similarity search. In *Proc. Int. Conf. Neural Inf. Process.* 256–267.
[49] Yongxin Wang, Xin Luo, Liqiang Nie, Jingkuan Song, Wei Zhang, and Xin Shun Xu. 2021. BATCH: A scalable asymmetric discrete cross-modal hashing. *IEEE Trans. Knowl. Data Eng.* (2021). https://doi.org/10.1109/TKDE.2020.2974825
[50] Yongxin Wang, Xin Luo, and Xin Shun Xu. 2020. Label embedding online hashing for cross-modal retrieval. In *Proc. ACM Multimedia Conf.* 871–879.
[51] Yunbo Wang, Xianfeng Ou, Jian Liang, and Zhenan Sun. 2021. Deep semantic reconstruction hashing for similarity retrieval. *IEEE Trans. Circuits Syst. Video Technol.* 31, 1 (2021), 387–400.
[52] Yair Weiss, Antonio Torralba, and Rob Fergus. 2009. Spectral hashing. In *Proc. Neural Inf. Process. Syst.* 1753–1760.
[53] Rongkai Xia, Yan Pan, Hanjiang Lai, Cong Liu, and Shuicheng Yan. 2014. Supervised hashing for image retrieval via image representation learning. In *Proc. AAAI Conf. Artif. Intell.* 2156–2162.
[54] Xing Xu, Fumin Shen, Yang Yang, Heng Tao Shen, and Xuelong Li. 2017. Learning discriminative binary codes for large-scale cross-modal retrieval. *IEEE Trans. Image Process.* 26, 5 (2017), 2494–2507.
[55] Jile Zhou, Guiguang Ding, and Yuchen Guo. 2014. Latent semantic sparse hashing for cross-modal similarity search. In *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.* 415–424.