Effective and Efficient Indexing in Cross-Modal Hashing-Based Datasets

Chih-Yi Chiu and Sarawut Markchit

Department of Computer Science and Information Engineering, National Chiayi University
No.300 Syuefu Rd., Chiayi City 60004, Taiwan
cychiu@mail.nctu.edu.tw; s1040465@mail.nctu.edu.tw
Phone: +886-5-2717228
ABSTRACT

To overcome the barrier of storage and computation, the hashing technique has been widely used for nearest neighbor search in multimedia retrieval applications recently. Particularly, cross-modal retrieval that searches across different modalities becomes an active but challenging problem. Although numerous of cross-modal hashing algorithms are proposed to yield compact binary codes, exhaustive search is impractical for large-scale datasets, and Hamming distance computation suffers inaccurate results. In this paper, we propose a novel search method that utilizes a probability-based index scheme over binary hash codes in cross-modal retrieval. The proposed hash code indexing scheme exploits a few binary bits of the hash code as the index code. We construct an inverted index table based on index codes, and train a neural network to improve the indexing accuracy and efficiency. Experiments are performed on two benchmark datasets for retrieval across image and text modalities, where hash codes are generated by five cross-modal hashing methods. Results show the proposed method effectively boosts the performance on search accuracy, computation cost, and memory consumption in these datasets and hashing methods. The source code is available on https://github.com/msarawut/HCI.

Index Terms – binary embedding; cross-modal retrieval; inverted indexing; learning to rank; nearest neighbor search.
1. INTRODUCTION

Nearest neighbor (NN) search plays a fundamental role in machine learning and information retrieval. Cross-modal retrieval, an application based on nearest neighbor search, has grabbed much research attention recently. It is natural that multimedia data have multiple modalities; these modalities may contribute correlated semantic information, such as video-tag pairs in YouTube and image-text pairs in Flickr. Cross-modal retrieval can return relevant results of one modality with respect to a given query of another modality. For example, we can use text queries to retrieve images, and use image queries to retrieve texts. This retrieval paradigm provides a useful and flexible interface for users to search for data across different modalities.

With the rapid growth of multimedia data, it is impractical to apply exhaustive search that consumes a tremendous computation resource in a large-scale dataset. Cross-modal hashing (CMH) that generates a compact data representation in the common space has become the main trend to overcome this problem. It embeds data points from the original space into a Hamming space as binary hash codes. A set of hash functions are learned by exploiting the inter/intra class correlation or underlying data distribution/manifold so that similar binary codes are generated for similar data points. Hamming distance computation between binary codes enables a fast nearest neighbor search through hardware-supported bit operations with least memory consumption. However, the use of hashing has to face the critical problem of quantization loss after binary embedding. Even though a number of learning-based hashing algorithms have been proposed to reduce the quantization loss, there exists an inevitable large information gap between real-valued vectors and the corresponding binary codes. Searching nearest neighbors among binary codes is therefore less accurate than that among real-valued vectors.
Existing CMH methods mainly put the focus on generating hash functions while pay little attention to the index scheme. Although Hamming distance computation and ranking is very efficient, exhaustive search over a large-scale binary code dataset is impractical for real-time applications. It is straightforward to construct an inverted index table based on binary hash codes. For example, the reference data points with the same binary code can be associated within a cluster and accessed through inverted indexing. However, indexing based on binary codes has the same inaccurate problem. Theoretically, increasing the number of bits of a binary code can enhance the discriminability, but a longer binary code takes more computation cost. Although multi-index hashing [21][23] that employs several index tables for Hamming distance computation is proposed to address the efficiency issue, it requires multiple times of memory more than single-index hashing. How to make a good tradeoff between search accuracy, computation efficiency, and memory consumption is still an open issue in cross-modal retrieval.

In this paper, we propose to utilize a novel index scheme over binary hash codes for cross-modal retrieval. The proposed index scheme exploits a few binary bits of the hash code as the index code. An index structure is built by compiling the reference data points with the same index codes into lists of an inverted table. Given a query, we estimate the relevance of each index code that implicitly reflects the probability distribution of nearest neighbors (ground truth) for the query. The estimation is realized by a prediction model that learns a nonlinear mapping between the query of one modality and the index space of another modality through the deep learning technique. Then we access the index table from the top rank index codes with the highest relevance scores to retrieve high quality candidates. We evaluate the proposed index scheme adopted on five state-of-the-art cross-modal hashing algorithms in two widely-used benchmark datasets. Experimental results show the proposed method can effectively improve the search performance in terms of retrieval accuracy, computation time, and memory consumption. The proposed method can be built upon any binary code datasets generated by
hashing algorithms to derive the following benefits:

- With the index structure built upon the binary hash codes, the retrieval process can only access partial reference data points with sub-linear time complexity, compared with the exhaustive search that takes linear time complexity to access all reference data points.
- Given a query, the learned prediction model can estimate the relevance scores of the index codes to provide precise indexing and obtain high quality candidates, rather than exhaustive search that ranks in the heavy-quantized Hamming space.
- By integrating the proposed index scheme, the supervised CMH methods can yield a better accuracy than the unsupervised ones, while the latter can achieve a close result with more efficient data access.
- Based on the precise indexing, our search algorithm can skip ranking by Hamming distances without performance degradation, and can further make the retrieval process more computation- and memory-efficient.

The remainder of this paper is organized as follows. Section 2 discusses the previous work about cross-modal hashing and retrieval. Section 3 presents the proposed probability-based index scheme and search algorithm. Section 4 shows experimental results. Conclusion remarks are given in Section 5.

2. RELATED WORK

Learning to hash has been a widely-accepted technique for similarity search in large-scale information retrieval. The goal of learning to hash is to represent the original data point by a binary code through a set of hash functions. With the hardware-supported machine instructions, the Hamming distance computation between two binary codes (in the binary space) can be
greatly accelerated compared to the Euclidean distance computation between two real-valued vectors (in the original space). The compact representation of binary codes not only enjoys the computation efficiency, but also lightens the memory loading.

The hashing technique can be classified into three main categories: uni-modal hashing, multi-view hashing, and CMH. Uni-modal hashing derives binary hash codes from a single type of features. The seminal works include locality-sensitive hashing [8] and iterative quantization [4], to name a few. However, conventional uni-modal hashing methods do not support multi-modal search well since similarities/distances cannot be directly computed across different modalities. Multi-view hashing utilizes multiple types of features simultaneously to learn a better binary code representation [15][24][27]. It requires that each modality should be observed in all data points, including the query and reference data. On the other hand, CMH usually embeds multiple heterogeneous data from different modalities into a common latent space where the discriminability or similarity correlation is preserved [22].

The CMH methods can be roughly divided into the unsupervised and supervised approaches dependent on that the class/label information is omitted or used in the learning process. In the unsupervised approach, composite correlation quantization (CCQ) [16] uses correlation-maximal mappings to transform data points from different modality types into an isomorphic latent space and then converts to binary codes by learning composite quantizers. Unsupervised generative adversarial hashing (UGACH) [29] exploits the generative adversarial network to train a generative model and a discriminative model. A correlation graph is used to capture the underlying manifold structure across different modalities. Multi-modal graph regularized smooth matrix factorization hashing (MSFH) [3] estimates the similarity of different modalities through a similarity graph and reduces the quantization loss in the hashing process. Robust and flexible discrete hashing (RFDH) [26] uses a discrete collective matrix factorization to generate unified hash codes for training data and takes classification models to learn hash functions. Fusion similarity hashing (FSH) [14] constructs an undirected asymmetric
graph to model the fusion similarity among different modalities and embeds the fusion similarity across modalities into a common Hamming space.

In the supervised approach, matrix factorization hashing [25] employs collective matrix factorization to generate unified hash codes. The label consistency across different modalities and local geometric consistency in each modality are applied to make the hash codes more discriminating. Supervised cross-modal hashing without relaxation (SCMH-WR) [6] learns a rotation matrix to minimize the quantization error for binary embedding. Discrete latent factor model (DLFH) [10] utilizes the discrete latent factor to model the supervised information and adopts the maximum likelihood loss function without relaxation. Supervised discrete manifold-embedded cross-modal hashing (SDMCH) [19] exploits the semantic information and the nonlinear manifold structure of data to construct the correlation among heterogeneous multiple modalities. Discrete latent semantic hashing (DLSH) [18] learns the latent semantic representations of different modalities and then projects them into the shared Hamming space. Discriminative correlation hashing (DCH) [17] introduces the linear discriminate analysis to preserve the discriminative property of one modality and transfers it to another modality. Semantics-preserving hashing (SePH) [13] considers the semantic consistency between multiple views. It transforms the given semantic affinities of training data to a probability distribution and approximates it with another one in the Hamming space via minimizing their KL divergence.

The deep neural network (DNN) technique has become the main trend in the supervised CMH approach. Deep cross-modal hashing (DCMH) [9] leverages DNN to learn hash functions based on a cross-modal similarity matrix that is defined by class labels. Deep discrete cross-modal hashing (DDCMH) [31] learns discrete nonlinear hash functions by preserving the intra-modality similarity at each hidden layer of the networks and the inter-modality similarity at the output layer of each individual network. Deep semantic-preserving ordinal hashing (DSPOH) [11] learns hash functions by exploiting the rank structure of feature dimensions. It
integrates the ranking-based hash learning, inter-modality similarity preserving and intra-modality class label learning into a unified deep network. Deep binary reconstruction (DBRC) [5] learns the binary hashing codes in an unsupervised fashion by the adaptive hyperbolic tangent function. Self-supervised adversarial hashing (SSAH) [12] employs two adversarial networks that maximize the semantic correlation and representation consistency between different modalities. Semi-supervised cross-modal hashing by generative adversarial network (SCH-GAN) [28] employs the generative model to select margin examples of one modality from unlabeled data for a query of another modality, while the discriminative model tries to distinguish the generated examples and true positive examples with respect to the query.

3. HASH CODE INDEXING

Given a reference dataset of binary codes, we construct an inverted index table based on the binary index codes. We train a prediction model that estimates the relevance scores of the index codes for a given query. From the view of CMH, the query is in one modality space (e.g., text or image) whereas the index codes are in the latent space (derived from text and image). The prediction model characterizes the mapping from the query space to the latent space and learns the neighborhood relation between the two spaces. The key to success is the training process for the prediction model, as elaborated in the following.

3.1. Index Model Construction and Training

Suppose that we have a reference dataset of $N$ binary codes of length $c$, denoted as $B = \{b_i \in \{0,1\}^c | i = 1, 2, ..., N\}$. The binary codes can be generated by any one of the CMH algorithms. We select $d$ binary bits from the binary code $b_i$ as its index code $x_i \in \{0, 1\}^d$. The binary bit selection is another research issue [1][20][30] but beyond the scope of this paper. Without loss of generality, we simply take the first $d$ bits as the index code. An index table with $2^d$ entries is
constructed based on index codes, where each entry $E_x = \{b_i \mid x_i = X\}$ represents a particular index code $X$ attaching a set of associated reference data points. We train a prediction model that learns a nonlinear mapping between the query of one modality (e.g., texts) and the index space of another modality (e.g., images) through deep learning. The model is then used to estimate the relevance scores of index codes for a given query. To compile the training dataset, we prepare a set of queries of one modality, denoted as $Q = \{q_j^\theta \mid j = 1, 2, ..., J\}$, where $q_j^\theta$ is the $j$th query. The relevant examples of another modality for $q_j^\theta$ are denoted as $\{b_{jk}^\theta \mid k = 1, 2, ..., K\} \in B$, where $b_{jk}^\theta$ is the $k$th relevant example for $q_j^\theta$. The definition of the relevant example is based on the class/label information with respect to the query. For example, the relevant examples of a text query are the images that have the same class to the query. The relevance score for each index code $X$ is defined by the proportion of relevant examples to the entry set size:

$$T_{jX}^\theta = \frac{|\{b_{jk}^\theta \mid x_{jk}^\theta = X\}|}{|E_X|},$$

(1)

where $|\cdot|$ denotes the set cardinality. The training set is compiled as pairs of query features and relevance scores; the $j$th query $q_j^\theta$ is associated with a set of $2^d$ relevance scores of index codes $\{T_{jX}^\theta\}$ as the target values.

The prediction model is characterized by a fully-connected neural network to learn the neighborhood relation between the query and index spaces based on the training set. The input layer receives the feature representation of $q_j^\theta$, and the output layer predicts $2^d$ relevance scores of index codes $\{P_{jX}\}$. Based on the loss between the predictions $\{P_{jX}\}$ and targets $\{T_{jX}^\theta\}$, we compute the error derivative with respect to the output of each unit, which is backward propagated to each layer in order to update the weights of the neural network.

3.2. Indexing and Retrieval

Given a query $q$ for cross-modal retrieval, we utilize the trained neural network to predict
the relevance scores of index codes \( \{P_X\} \). Let \( f \) be the prediction model expressed as:

\[
f(q) = \{P_X\},
\]

which maps the query feature to the relevance score distribution over the index codes. The index codes are ranked to select the top-\( R \) index codes \( \{X_1, X_2, ..., X_R\} \) with the highest relevance scores:

\[
\{X_1, X_2, ..., X_R\} = \text{arg max}_{X} (P_X, R),
\]

where \( X_r \) is index code with the \( r \)th highest relevance score, and function \( \text{arg max}_{X} (P_X, R) \) returns the \( R \) arguments indexed by \( X \) for \( R \) minimums in \( \{P_X\} \). The reference data points associated with the top-ranking index codes are included in a candidate set \( C \) until \( k \) candidates are obtained. An alternative way is that we collect more candidates to be further examined for reranking. That is, we compute the Hamming distances between the query and all candidates, and then sort the candidates based on their Hamming distances to output the first \( k \) nearest neighbors. Since the size of the candidate set is usually a small fraction of the reference dataset, it takes a few computation overheads for reranking rather than exhaustive search. However, the retrieval accuracy may not be improved if the reranking is based on the inaccurate Hamming distances. According to our empirical studies, reranking in the Hamming space is unnecessary. We may skip the reranking step to simplify the whole retrieval process for acceleration without sacrificing accuracy. Moreover, we do not have to keep the reference dataset for the reranking purpose, so the memory consumption is reduced. Algorithm 1 summarizes the index and retrieval process.
**Algorithm 1**: Indexing and Retrieval

**Input**: index table \( \{E_X\} \), prediction model \( f \), query \( q \);

**Output**: \( k \) nearest neighbors of \( q \);

1. Predict the relevance scores of all index codes \( f(q) = \{P_X\} \);
2. Select the top-\( R \) index codes \( \{X_1, X_2, ..., X_R\} = \arg\max_X(P_X, R) \);
3. Initialize an empty candidate set \( C = \emptyset \) and a parameter \( r = 1 \);
4. If \( |C| > k \), go to Step 7;
5. Include the reference data points indexed in \( X_r \) into \( C = C \cup E_{X_r} \);
6. Set \( r = r + 1 \) and go to Step 4;
7. Output \( C \) as the nearest neighbors of \( q \);

### 3.3. Complexity Analysis

We give the time and space complexity analysis for Algorithm 1 in the following. The time complexity mainly involves two parts, namely, the relevance score prediction and index code ranking. The time spent for relevance score prediction is related to the size of the neural network; it can be accelerated by GPUs in constant time. Suppose the network has \( \tau \) hidden layers, each of which contains \( \lambda \) units, and the last output layer has \( 2^d \) units corresponding to \( 2^d \) index entries. The prediction spends \( O(\tau \cdot \lambda^2 + \lambda \cdot 2^d) \) time. Index code ranking requires to sort all index codes based on their relevance scores; it takes \( O(2^d \cdot \log 2^d) = O(d \cdot 2^d) \) time. Note that we do not rerank the candidates based on their Hamming distances to the query; the quality of the candidate set is good enough to be the output result, as demonstrated in the experimental section. Therefore, the time complexity is further reduced by removing Hamming distance reranking.

The memory space consumed by Algorithm 1 is the whole index structure, which must be loaded into memory for real-time response in search applications. The index structure size is determined by the number of data points indexed in the index table, index entry codebook, and prediction neural network. Assume the number of reference data points \( N \) is no more than \( 2^{32} \), we can use 4 bytes to encode the data identity, and \( 4N \) bytes are required for \( N \) reference data.
points. An index table associates data identities through 64-bit pointers for inverted indexing, which take $8 \cdot 2^d$ bytes for a total of $2^d$ index entries. The index entry codebook size is $\frac{d \cdot 2^d}{8}$ bytes, where each index entry occupies $\frac{d}{8}$ bytes. The prediction neural network requires $4(\tau \cdot \lambda^2 + \lambda \cdot 2^d)$ bytes, where each network coefficient is represented by a 4-byte floating-point number.

4. EXPERIMENTS

To evaluate the proposed method, the experiment is conducted by using five state-of-the-art CMH methods on two widely-used benchmark datasets. The benchmark datasets are MIRFlickr [7] and NUSWIDE [2], each of which consists of an image modality and a text modality. The CMH methods, including DLFH [10], SePH [13], DCMH [9], CCQ [16] and FSH [14], are employed to generate binary code datasets for MIRFlickr and NUS-WIDE individually, denoted as CMH datasets. The experiments are performed on a PC with Intel i7 CPU@3.6 GHz and 32GB RAM.

4.1. CMH datasets

Table 1 summarizes the properties of the two benchmark datasets, which are then used to produce the CMH datasets. The original MIRFlickr dataset has 25000 instances collected from the Flickr website. Each instance consists of an image, associated textual tag, and one or more of 24 predefined semantic labels. We removed textual tags that appear less than 20 times in the dataset, and then deleted instances that without any textual tag or semantic label. For each instance, its image view is characterized by a 150-D edge histogram, and its text view is represented as a 500-D feature vector derived from PCA on its binary tagging vector with respect to the textual tags. We took 5% of MIRFlickr data to form the query set and the rest as
the reference set. 10000 instances were sampled from the reference set for training. The ground-truth neighbors were defined as those image-text pairs which share at least one common label.

For the original NUS-WIDE dataset, it has 260648 instances, each of which consists of an image and one or more of 81 predefined semantic labels. We selected 195834 image-text pairs that belong to the 21 most frequent concepts. The text for each point is represented as a 1000-dimensional bag-of-word vector. The hand-crafted feature for each image is a 500-dimensional bag-of-visual word (BOVW) vector. We used 2000 data points as the query set and the remaining points as the reference set. 20000 data points were sampled from the reference set for training. The ground truth neighbors were defined as those image-text pairs which share at least one common label, as the same to MIRFlickr.

We selected five state-of-the-art CMH methods to generate binary codes. These methods included the supervised approach DLFH [10], SePH [13], and DCMH [9], and the unsupervised approach CCQ [17] and FSH [15], each of which generated 16-, 32-, and 64-bit binary codes for MIRFlickr and NUS-WIDE datasets. In total thirty CMH datasets were produced and evaluated in the experiment.

| Table 1. Benchmark datasets |
|-----------------------------|
| MIRFlickr | NUS-WIDE |
| Dataset size | 16738 | 195834 |
| Reference set | 15902 | 193834 |
| Training set | 10000 | 20000 |
| Query set | 836 | 2000 |
| Number of Labels | 24 | 21 |
4.2. Implementation and Comparison

Three kinds of index schemes are implemented for comparison:

- **Exhaustive.** It calculates Hamming distances between the query and all reference data without adopting any index scheme. The five CMH methods applying the exhaustive search scheme are served as the baselines.

- **Naïve-index (d bits).** It takes the first $d$ bits of the hash code as the index code for clustering reference data points. The given query compares the index code to find candidates and then rerank the candidates according to their Hamming distances. Here $d = 14$.

- **DNN-index (d bits).** It is the proposed method. In addition to the naive index structure, we learn a 4-layer fully-connected neural network as the prediction model for index codes. Table 2 lists the network configurations for the five CMH methods. Here $d \in \{8, 10, 12, 14\}$. 

| Layer       | The number of units | Activation function |
|-------------|---------------------|---------------------|
| Input       | Query raw data $q$  | ReLU                |
| Hidden      | 8                   | 32                  | 8       | 16      | 8       | ReLU                |
| Hidden      | 16                  | 64                  | 16      | 32      | 16      | ReLU                |
| Hidden      | 8                   | 32                  | 8       | 16      | 8       | ReLU                |
| Output      | The number of clusters $2^d$ | Softmax             |

Mean average precision (MAP) is used to evaluate the retrieval accuracy for a set of queries $Q$:

$$ MAP@R = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{R} \sum_{j=1}^R pr(j).rel(j), $$

(4)
where \( R \) is the number of retrieved documents, \( pr(j) \) denotes the precision of the top \( j \) retrieved examples, and \( rel(j) = 1 \) if the \( j \)th retrieved example is relevant to the query, otherwise \( rel(j) = 0 \). MAP is computed as the mean of all queries’ average precision. The source code is available on https://github.com/msarawut/HCI.

4.3. Results and Discussions

Figures 1 and 2 show the results for the search modality "text query vs. image dataset" (\( T \rightarrow I \)) in MIRFlickr and NUS-WIDE datasets, respectively. Figures 3 and 4 demonstrate another search modality "image query vs. text dataset" (\( I \rightarrow T \)). Each row shows the subfigures of 16-bit, 32-bit, and 64-bit binary code datasets with respect to a particular CMH method. The X-axis and Y-axis represent the number of retrieved examples \( R \) and MAP@\( R \), respectively. In most cases, the exhaustive search and the naïve index schemes had similar MAP curves. The latter only took a few candidates for reranking but still reached a comparable accuracy to the former. The former did not benefit from reranking all reference data since the Hamming distances to the query have serious quantization loss and did not reflect the actual similarities well.
Figure 1. MAP in the MIRFlickr dataset for text query vs. image dataset.
Figure 2. MAP in the NUS-WIDE dataset for text query vs. image dataset.
Figure 3. MAP in the MIRFlickr dataset for image query vs. text dataset.
Figure 4. MAP in the NUS-WIDE dataset for image query vs. text dataset.
We observe that the DNN-index scheme with long index bits generated similar MAP curves across various reference datasets. For example, in Figure 1, the 14-bit DNN index scheme generally performed stably in all subfigures, despite of binary hash code lengths and CMH methods. In addition, the longer index code can generate a more compact candidate list. Tables 3 and 4 compare our method with these CMH algorithms for $T \rightarrow I$ and $I \rightarrow T$, respectively, in terms of MAP@50 and the fraction of accessed reference data (ARD%), which is defined by:

$$\text{ARD}\% = \frac{\text{The number of candidates}}{\text{The number of reference data points}} \times 100\%$$  \hspace{1cm} (4)

A lower ARD% means a smaller computation cost due to less memory access operations for the reference data. The 14-bit DNN index scheme, which obtained the highest accuracy and smallest computation cost, showed a significant improvement when it integrated with these CMH methods. The best results are shown in boldface.

A common sense is that in a higher dimensionality, the distance computation can be more discriminative to provide precise ranking. However, it is not always this case when in the Hamming space. This observation again indicates the instability of ranking based on Hamming distances. Interestingly, although the unsupervised CMH methods (i.e., CCQ and FSH) are not as accurate as the supervised CMH methods (i.e., DLFH, SePH, and DCMH), integrating the proposed DNN-index scheme can effectively boost the accuracy for all CMH methods, and the unsupervised CMH methods became comparable to the supervised. Moreover, with the DNN-index scheme, the unsupervised CMH methods yielded smaller ARD% rates than the supervised ones. This is because the unsupervised CMH methods produced a more balanced data distribution among index clusters so the data access is more efficient. On the other hand, the supervised CMH methods obtained higher MAP rates since the class information is embedded in binary hash codes.
Table 3. Comparison in terms of MAP@50 and ARD% in the MIRFlickr dataset

| Task | Methods                | 16-bit |          | 32-bit |          | 64-bit |          |
|------|------------------------|--------|----------|--------|----------|--------|----------|
|      |                        | MAP@50 | ARD%     | MAP@50 | ARD%     | MAP@50 | ARD%     |
|      |                        |        |          |        |          |        |          |
| T→I  | DLFH                   | 0.8529 | 100%     | 0.8887 | 100%     | 0.9133 | 100%     |
|      | SePH                   | 0.7137 | 100%     | 0.7493 | 100%     | 0.7761 | 100%     |
|      | DCMH                   | 0.7451 | 100%     | 0.7660 | 100%     | 0.7852 | 100%     |
|      | CCQ                    | 0.4842 | 100%     | 0.4539 | 100%     | 0.4086 | 100%     |
|      | FSH                    | 0.4636 | 100%     | 0.4851 | 100%     | 0.5042 | 100%     |
|      | DLFH DNN-index (8 bits)| 0.8670 | 1.75%    | 0.8700 | 1.75%    | 0.8736 | 1.75%    |
|      | SePH DNN-index (8 bits)| 0.7289 | 3.03%    | 0.7383 | 3.03%    | 0.7481 | 3.03%    |
|      | DCMH DNN-index (8 bits)| 0.7650 | 4.40%    | 0.7686 | 4.40%    | 0.7742 | 4.40%    |
|      | CCQ DNN-index (8 bits) | 0.7194 | 0.90%    | 0.7176 | 0.90%    | 0.7173 | 0.90%    |
|      | FSH DNN-index (8 bits) | 0.5622 | 1.96%    | 0.5647 | 1.96%    | 0.5669 | 1.96%    |
|      | DLFH DNN-index (14 bits)| 0.9146 | 1.08%    | 0.9151 | 1.08%    | 0.9158 | 1.08%    |
|      | SePH DNN-index (14 bits)| 0.8436 | 0.34%    | 0.8440 | 0.34%    | 0.8441 | 0.34%    |
|      | DCMH DNN-index (14 bits)| 0.8080 | 1.26%    | 0.8105 | 1.26%    | 0.8118 | 1.26%    |
|      | CCQ DNN-index (14 bits)| 0.8753 | 0.33%    | 0.8754 | 0.33%    | 0.8754 | 0.33%    |
|      | FSH DNN-index (14 bits)| 0.8527 | 1.28%    | 0.8527 | 1.28%    | 0.8527 | 1.28%    |
| I→T  | DLFH                   | 0.8160 | 100%     | 0.8283 | 100%     | 0.8563 | 100%     |
|      | SePH                   | 0.5992 | 100%     | 0.6179 | 100%     | 0.6274 | 100%     |
|      | DCMH                   | 0.6899 | 100%     | 0.7075 | 100%     | 0.7359 | 100%     |
|      | CCQ                    | 0.4011 | 100%     | 0.3996 | 100%     | 0.3828 | 100%     |
|      | FSH                    | 0.4887 | 100%     | 0.5073 | 100%     | 0.5321 | 100%     |
|      | DLFH DNN-index (8 bits)| 0.8774 | 0.94%    | 0.8799 | 0.94%    | 0.8819 | 0.94%    |
|      | SePH DNN-index (8 bits)| 0.6958 | 3.09%    | 0.6988 | 3.09%    | 0.7016 | 3.09%    |
|      | DCMH DNN-index (8 bits)| 0.6769 | 1.04%    | 0.6799 | 1.04%    | 0.6831 | 1.04%    |
|      | CCQ DNN-index (8 bits) | 0.6613 | 5.33%    | 0.6608 | 5.33%    | 0.6619 | 5.33%    |
|      | FSH DNN-index (8 bits) | 0.5649 | 1.64%    | 0.5682 | 1.64%    | 0.5748 | 1.64%    |
|      | DLFH DNN-index (14 bits)| 0.9063 | 0.55%    | 0.9071 | 0.55%    | 0.9080 | 0.55%    |
|      | SePH DNN-index (14 bits)| 0.8459 | 0.33%    | 0.8460 | 0.33%    | 0.8460 | 0.33%    |
|      | DCMH DNN-index (14 bits)| 0.8221 | 0.28%    | 0.8221 | 0.28%    | 0.8221 | 0.28%    |
|      | CCQ DNN-index (14 bits)| 0.8803 | 0.32%    | 0.8803 | 0.32%    | 0.8803 | 0.32%    |
|      | FSH DNN-index (14 bits)| 0.8534 | 0.28%    | 0.8534 | 0.28%    | 0.8534 | 0.28%    |
Table 4. Comparison in terms of MAP@50 and ARD% in the NUS-WIDE dataset

| Task | Methods | 16-bit | 32-bit | 64-bit |
|------|---------|--------|--------|--------|
|      |         | MAP@50 | ARD%   | MAP@50 | ARD%   | MAP@50 | ARD%   |
|      | T→I     |        |        |        |        |        |        |
|      | DLFH    | 0.8457 | 100%   | 0.8418 | 100%   | 0.8448 | 100%   |
|      | SePH    | 0.5303 | 100%   | 0.5992 | 100%   | 0.6597 | 100%   |
|      | DCMH    | 0.5777 | 100%   | 0.5961 | 100%   | 0.6126 | 100%   |
|      | CCQ     | 0.1666 | 100%   | 0.1565 | 100%   | 0.1488 | 100%   |
|      | FSH     | 0.4337 | 100%   | 0.2861 | 100%   | 0.4826 | 100%   |
|      | DLFH DNN-index (8 bits) | 0.8738 | 0.41% | 0.8769 | 0.41% | 0.8767 | 0.41% |
|      | SePH DNN-index (8 bits) | 0.5057 | 10.23% | 0.5581 | 10.23% | 0.5610 | 10.23% |
|      | DCMH DNN-index (8 bits) | 0.6329 | 5.38% | 0.6278 | 5.38% | 0.6312 | 5.38% |
|      | CCQ DNN-index (8 bits) | 0.4283 | 0.53% | 0.4289 | 0.53% | 0.4319 | 0.53% |
|      | FSH DNN-index (8 bits) | 0.4379 | 8.12% | 0.4399 | 8.12% | 0.4394 | 8.12% |
|      | DLFH DNN-index (14 bits) | **0.8788** | 0.19% | **0.8790** | 0.19% | **0.8792** | 0.19% |
|      | SePH DNN-index (14 bits) | 0.6998 | 0.03% | 0.7001 | 0.03% | 0.7002 | 0.03% |
|      | DCMH DNN-index (14 bits) | 0.7564 | 0.23% | 0.7583 | 0.23% | 0.7581 | 0.23% |
|      | CCQ DNN-index (14 bits) | 0.7289 | 0.03% | 0.7288 | 0.03% | 0.7288 | 0.03% |
|      | FSH DNN-index (14 bits) | 0.6593 | **0.03%** | 0.6594 | **0.03%** | 0.6598 | **0.03%** |
|      | I→T     |        |        |        |        |        |        |
|      | DLFH    | 0.8471 | 100%   | 0.7881 | 100%   | 0.7289 | 100%   |
|      | SePH    | 0.3747 | 100%   | 0.4037 | 100%   | 0.4404 | 100%   |
|      | DCMH    | 0.4823 | 100%   | 0.6005 | 100%   | 0.5679 | 100%   |
|      | CCQ     | 0.1601 | 100%   | 0.1530 | 100%   | 0.1453 | 100%   |
|      | FSH     | 0.4261 | 100%   | 0.2920 | 100%   | 0.4259 | 100%   |
|      | DLFH DNN-index (8 bits) | 0.8148 | 0.50% | 0.8168 | 0.50% | 0.8174 | 0.50% |
|      | SePH DNN-index (8 bits) | 0.3701 | 10.47% | 0.3937 | 10.47% | 0.3941 | 10.47% |
|      | DCMH DNN-index (8 bits) | 0.4022 | 1.69% | 0.4136 | 1.69% | 0.4394 | 1.69% |
|      | CCQ DNN-index (8 bits) | 0.3467 | 0.44% | 0.3418 | 0.44% | 0.3420 | 0.44% |
|      | FSH DNN-index (8 bits) | 0.2942 | 7.47% | 0.2876 | 7.47% | 0.2899 | 7.47% |
|      | DLFH DNN-index (14 bits) | **0.9126** | 0.05% | **0.9127** | 0.05% | **0.9127** | 0.05% |
|      | SePH DNN-index (14 bits) | 0.6305 | 0.06% | 0.6308 | 0.06% | 0.6307 | 0.06% |
|      | DCMH DNN-index (14 bits) | 0.6775 | 0.03% | 0.6775 | 0.03% | 0.6775 | 0.03% |
|      | CCQ DNN-index (14 bits) | 0.6358 | 0.03% | 0.6358 | 0.03% | 0.6358 | 0.03% |
|      | FSH DNN-index (14 bits) | 0.6373 | **0.03%** | 0.6374 | **0.03%** | 0.6373 | **0.03%** |
To investigate the effect on Hamming distance reranking, we evaluated two different ranking approaches after DNN-indexing: one applied Hamming distance reranking (HDR) and the other did not (no-HDR). Figures 5, 6, 7, and 8 show the comparison in 64-bit MIRFlickr and NUS-WIDE datasets for $T\rightarrow I$ and $I\rightarrow T$ tasks. Note that the no-HDR approach only ranked based on cluster relevance scores. We kept the first $R$ documents if the retrieved clusters had more than $R$ documents and these retrieved documents were ranked randomly. On the other hand, the HDR approach first ranked clusters then sorted retrieved documents based on their Hamming distances to output top $R$ documents. The results show that when no-HDR was applied, the performance did not degrade too much; it performed even slightly better in some cases. We consider it is unnecessary to perform reranking based on imprecise Hamming distances after the accurate DNN-indexing. In addition, we do not have to keep the reference binary code dataset for reranking. Both computation cost and memory consumption can be thus reduced.

Table 5 compares all baselines and the proposed method in MIRFlickr and NUS-WIDE datasets in terms of MAP@50, runtime, and memory consumption. The DLFH DNN-index scheme that yielded the highest MAP was used as the representative of the proposed method, and no Hamming distance reranking was applied. The average runtime was measured in milliseconds per query, and the memory consumption was measured in bytes per data point and omitted the 4-byte data identity. The memory consumption for the baselines comes from the storage for the reference binary codes, whereas that for the proposed method comes from the use of the index structure, as we analyzed in Section 3.3. In addition to the accuracy superiority, the proposed method provides more computation-efficient and memory-efficient retrieval in a larger-scale dataset, compared with the baselines.
Figure 5. Comparison of Hamming distance reranking (HDR) and without HDR (no-HDR) in the MIRFlickr dataset for text query vs. image dataset.

Figure 6. Comparison of Hamming distance reranking (HDR) and without HDR (no-HDR) in the NUS-WIDE dataset for text query vs. image dataset.
Figure 7. Comparison of Hamming distance reranking (HDR) and without HDR (no-HDR) in the MIRFlickr dataset for image query vs. text dataset.

Figure 8. Comparison of Hamming distance reranking (HDR) and without HDR (no-HDR) in the NUS-WIDE dataset for image query vs. text dataset.
Table 5. Comparison in terms of MAP@50, runtime (milliseconds per query), and memory consumption (bytes per data point)

| Task | Methods         | 16-bit  | 32-bit  | 64-bit  |
|------|----------------|---------|---------|---------|
|      |                | MAP@50  | time    | mem.    | MAP@50  | time    | mem.    | MAP@50  | time    | mem.    |
|      |                |         |         |         |         |         |         |         |         |         |
|      | MIRFlickr (15902 data points) |         |         |         |         |         |         |         |         |         |
|      | T→I | DLFH  | 0.8529 | 1.46   | 2.0    | 0.8887 | 2.32   | 4.0    | 0.9133 | 3.87   | 8.0    |
|      |     | SePH  | 0.7137 | 1.72   | 2.0    | 0.7493 | 2.63   | 4.0    | 0.7761 | 4.69   | 8.0    |
|      |     | DCMH  | 0.7451 | 2.01   | 2.0    | 0.7660 | 2.84   | 4.0    | 0.7852 | 5.03   | 8.0    |
|      |     | CCQ   | 0.4842 | 2.19   | 2.0    | 0.4539 | 3.53   | 4.0    | 0.4086 | 6.42   | 8.0    |
|      |     | FSH   | 0.4636 | 2.32   | 2.0    | 0.4851 | 3.87   | 4.0    | 0.5042 | 6.74   | 8.0    |
|      |     | DNN-index (8 bits) | 0.8603 | 0.34   | 1.9    | 0.8603 | 0.34   | 1.9    | 0.8603 | 0.34   | 1.9    |
|      |     | DNN-index (14 bits) | 0.9147 | 1.04   | 52.6   | 0.9147 | 1.04   | 52.6   | 0.9147 | 1.04   | 52.6   |
|      | I→T | DLFH  | 0.8160 | 1.47   | 2.0    | 0.8283 | 2.17   | 4.0    | 0.8563 | 3.84   | 8.0    |
|      |     | SePH  | 0.5992 | 1.73   | 2.0    | 0.6179 | 2.62   | 4.0    | 0.6274 | 4.95   | 8.0    |
|      |     | DCMH  | 0.6899 | 2.07   | 2.0    | 0.7075 | 3.15   | 4.0    | 0.7359 | 5.84   | 8.0    |
|      |     | CCQ   | 0.4011 | 2.17   | 2.0    | 0.3996 | 3.81   | 4.0    | 0.3828 | 6.58   | 8.0    |
|      |     | FSH   | 0.4887 | 2.43   | 2.0    | 0.5073 | 3.92   | 4.0    | 0.5321 | 7.01   | 8.0    |
|      |     | DNN-index (8 bits) | 0.8490 | 0.19   | 1.9    | 0.8490 | 0.19   | 1.9    | 0.8490 | 0.19   | 1.9    |
|      |     | DNN-index (14 bits) | 0.9021 | 0.93   | 52.6   | 0.9021 | 0.93   | 52.6   | 0.9021 | 0.93   | 52.6   |
|      | NUS-WIDE (193834 data points) |         |         |         |         |         |         |         |         |         |
|      | T→I | DLFH  | 0.8457 | 13.90  | 2.0    | 0.8418 | 19.45  | 4.0    | 0.8448 | 32.65  | 8.0    |
|      |     | SePH  | 0.5303 | 13.33  | 2.0    | 0.5992 | 20.83  | 4.0    | 0.6597 | 36.38  | 8.0    |
|      |     | DCMH  | 0.5777 | 13.79  | 2.0    | 0.5961 | 20.65  | 4.0    | 0.6126 | 35.96  | 8.0    |
|      |     | CCQ   | 0.1666 | 18.85  | 2.0    | 0.1565 | 28.35  | 4.0    | 0.1488 | 31.36  | 8.0    |
|      |     | FSH   | 0.7337 | 11.44  | 2.0    | 0.2861 | 16.72  | 4.0    | 0.4826 | 23.48  | 8.0    |
|      |     | DNN-index (8 bits) | 0.8686 | 0.62   | 0.2    | 0.8686 | 0.62   | 0.2    | 0.8686 | 0.62   | 0.2    |
|      |     | DNN-index (14 bits) | 0.8780 | 1.27   | 4.4    | 0.8780 | 1.27   | 4.4    | 0.8780 | 1.27   | 4.4    |
|      | I→T | DLFH  | 0.7289 | 13.89  | 2.0    | 0.7881 | 19.47  | 4.0    | 0.8471 | 34.03  | 8.0    |
|      |     | SePH  | 0.3747 | 13.29  | 2.0    | 0.4037 | 20.54  | 4.0    | 0.4404 | 36.67  | 8.0    |
|      |     | DCMH  | 0.4823 | 15.79  | 2.0    | 0.6005 | 24.20  | 4.0    | 0.5679 | 42.42  | 8.0    |
|      |     | CCQ   | 0.1601 | 18.92  | 2.0    | 0.1530 | 27.59  | 4.0    | 0.1453 | 31.66  | 8.0    |
|      |     | FSH   | 0.4261 | 10.96  | 2.0    | 0.2920 | 14.00  | 4.0    | 0.4259 | 21.96  | 8.0    |
|      |     | DNN-index (8 bits) | 0.8241 | 0.65   | 0.2    | 0.8241 | 0.65   | 0.2    | 0.8241 | 0.65   | 0.2    |
|      |     | DNN-index (14 bits) | 0.9095 | 1.02   | 4.4    | 0.9095 | 1.02   | 4.4    | 0.9095 | 1.02   | 4.4    |
5. CONCLUSION

In this paper, we propose a novel search method that utilizes a probability-based index scheme over binary hash codes in cross-modal retrieval. The index scheme ranks the hash index codes of the inverted table through DNN. It not only effectively improves the search accuracy but also efficiently reduce the computation and memory cost. Extensive experimental results show the outperformance of the proposed method compared with several state-of-the-art CMH methods in MIRFlickr and NUS-WIDE datasets.

ACKNOWLEDGEMENT

This work was supported by the Ministry of Science and Technology, Taiwan, under grants MOST 106-2221-E-415-019-MY3.

REFERENCES

[1] C. Y. Chiu, Y. C. Liou, A. Prayoonwong, Approximate asymmetric search for binary embedding codes, ACM Transactions on Multimedia Computing, Communications, and Applications. 2017 Jan;13(1): 3:1-25.

[2] T. S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, Y. Zheng, NUS-WIDE: a real-world web image database from National University of Singapore, In Proceedings of ACM International Conference on Image and Video Retrieval 2009 Jul 8 (p. 48). ACM.

[3] Y. Fang, H. Zhang, Y. Ren, Unsupervised cross-modal retrieval via multi-modal graph regularized smooth matrix factorization hashing, Knowledge-Based Systems. 2019 May 1;171:69-80.

[4] Y. Gong, S. Lazebnik, A. Gordo, F. Perronnin, Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval, IEEE Transactions on Pattern Analysis and Machine Intelligence. 2013 Dec;35(12):2916-29.
[5] D. Hu, F. Nie, X. Li, Deep binary reconstruction for cross-modal hashing, *IEEE Transactions on Multimedia*. 2018 Aug 27.

[6] H. J. Huang, R. Yang, C. X. Li, Y. Shi, S. Guo, X. S. Xu, Supervised cross-modal hashing without relaxation, In *IEEE International Conference on Multimedia and Expo (ICME)* 2017 Jul 10 (pp. 1159-1164). IEEE.

[7] M. J. Huiskes, M. S. Lew, The MIR Flickr retrieval evaluation, In *Proceedings of ACM International Conference on Multimedia Information Retrieval* 2008 Oct 30 (pp. 39-43). ACM.

[8] P. Indyk, R. Motwani, Approximate nearest neighbors: towards removing the curse of dimensionality, In *Proceedings of ACM Symposium on Theory of Computing* 1998 May 23 (pp. 604-613). ACM.

[9] Q. Y. Jiang, W. J. Li, Deep cross-modal hashing, In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* 2017 (pp. 3232-3240).

[10] Q. Y. Jiang, W. J. Li, Discrete latent factor model for cross-modal hashing, *IEEE Transactions on Image Processing*. 2019 Feb 6. (online available)

[11] L. Jin, K. Li, Z. Li, F. Xiao, G. J. Qi, J. Tang, Deep semantic-preserving ordinal hashing for cross-modal similarity search, *IEEE Transactions on Neural Networks and Learning Systems*. 2018 Oct 1. (online available)

[12] C. Li, C. Deng, N. Li, W. Liu, X. Gao, D. Tao, Self-supervised adversarial hashing networks for cross-modal retrieval, In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* 2018 (pp. 4242-4251).

[13] Z. Lin, G. Ding, J. Han, J. Wang, Cross-view retrieval via probability-based semantics-preserving hashing, *IEEE Transactions on Cybernetics*. 2017 Dec;47(12):4342-55.

[14] H. Liu, R. Ji, Y. Wu, F. Huang, B. Zhang, Cross-modality binary code learning via fusion similarity hashing, In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* 2017 (pp. 7380-7388).

[15] L. Liu, M. Yu, L. Shao, Multiview alignment hashing for efficient image search, *IEEE Transactions on Image Processing*. 2015 Mar;24(3):956-66.

[16] M. Long, Y. Cao, J. Wang, P. S. Yu, Composite correlation quantization for efficient multimodal retrieval, In *Proceedings of International ACM SIGIR Conference on Research and Development in Information Retrieval* 2016 Jul 7 (pp. 579-588). ACM.
[17] X. Lu, H. Zhang, J. Sun, Z. Wang, P. Guo, W. Wan, Discriminative correlation hashing for supervised cross-modal retrieval, Signal Processing: Image Communication. 2018 Jul 1;65:221-30.

[18] X. Lu, L. Zhu, Z. Cheng, X. Song, H. Zhang, Efficient discrete latent semantic hashing for scalable cross-modal retrieval, Signal Processing. 2019 Jan 1;154:217-31.

[19] X. Luo, X. Y. Yin, L. Nie, X. Song, Y. Wang, X. S. Xu, SDMCH: Supervised discrete manifold-embedded cross-modal hashing, In proceedings of International Joint Conference on Artificial Intelligence (IJCAI-18) 2018 Jul 13 (pp. 2518-2524).

[20] Y. Ma, H. Xie, Z. Chen, Q. Dai, Y. Huang, G. Ji, Fast search of binary codes with distinctive bits, In Proceedings of Pacific-Rim Conference on Multimedia (PCM) 2014.

[21] M. Norouzi, A. Punjani, D. J. Fleet, Fast exact search in hamming space with multi-index hashing, IEEE Transactions on Pattern Analysis and Machine Intelligence. 2014 Jun;36(6):1107-1119.

[22] Y. Peng, X. Huang, Y. Zhao, An overview of cross-media retrieval: concepts, methodologies, benchmarks, and challenges, IEEE Transactions on Circuits and Systems for Video Technology. 2018 Sep;28(9):2372-85.

[23] J. Song, H. T. Shen, J. Wang, N. Sebe, J. Wang, A distance-computation-free search scheme for binary code databases, IEEE Transactions on Multimedia. 2016 Mar;18(3):484-495.

[24] J. Song, Y. Yang, Z. Huang, H. T. Shen, J. Luo, Effective multiple feature hashing for large-scale near-duplicate video retrieval, IEEE Transactions on Multimedia. 2013 Dec;15(8):1997-2008.

[25] J. Tang, K. Wang, L. Shao, Supervised matrix factorization hashing for cross-modal retrieval, IEEE Transactions on Image Processing. 2016 Jul;25(7):3157-66.

[26] D. Wang, Q. Wang, X. Gao, Robust and flexible discrete hashing for cross-modal similarity search, IEEE Transactions on Circuits and Systems for Video Technology. 2018 Oct;28(10):2703-15.

[27] D. Zhang, F. Wang, L. Si, Composite hashing with multiple information sources, In Proceedings of International ACM SIGIR Conference on Research and Development in Information Retrieval 2011 Jul 24 (pp. 225-234). ACM.

[28] J. Zhang, Y. Peng, M. Yuan, SCH-GAN: Semi-supervised cross-modal hashing by generative adversarial network, IEEE Transactions on Cybernetics. 2018 Sep 26. (online available)
[29] J. Zhang, Y. Peng, M. Yuan, Unsupervised generative adversarial cross-modal hashing. In *AAAI Conference on Artificial Intelligence* 2018 Apr 25.

[30] X. Zhang, J. Qin, W. Wang, Y. Sun, J. Lu, HmSearch: An efficient hamming distance query processing algorithm. In *Proceedings of International Conference on Scientific and Statistical Database Management (SSDBM)* 2013.

[31] F. Zhong, Z. Chen, G. Min, Deep discrete cross-modal hashing for cross-media retrieval. *Pattern Recognition*. 2018 Nov 1;83:64-77.