Multi-View Complementary Hash Tables for Nearest Neighbor Search

Xianglong Liu[†] Lei Huang[†] Cheng Deng[‡] Jiwen Lu [♮] Bo Lang[†]
[†]State Key Lab of Software Development Environment, Beihang University, Beijing, China
[‡]Xidian University, Xi'an, China
[♮]Department of Automation, Tsinghua University, Beijing, China
{xlliu, huanglei, langbo}@nlsde.buaa.edu.cn chdeng.xd@gmail.com elujiwen@gmail.com

Abstract

*Recent years have witnessed the success of hashing techniques in fast nearest neighbor search. In practice many applications (*e.g.*, visual search, object detection, image matching, etc.) have enjoyed the benefits of complementary hash tables and information fusion over multiple views. However, most of prior research mainly focused on compact hash code cleaning, and rare work studies how to build multiple complementary hash tables, much less to adaptively integrate information stemming from multiple views. In this paper we first present a novel multi-view complementary hash table method that learns complementary hash tables from the data with multiple views. For single multi-view table, using exemplar based feature fusion, we approximate the inherent data similarities with a low-rank matrix, and learn discriminative hash functions in an efficient way. To build complementary tables and meanwhile maintain scalable training and fast out-of-sample extension, an exemplar reweighting scheme is introduced to update the induced low-rank similarity in the sequential table construction framework, which indeed brings mutual benefits between tables by placing greater importance on exemplars shared by mis-separated neighbors. Extensive experiments on three large-scale image datasets demonstrate that the proposed method significantly outperforms various naive solutions and state-of-the-art multi-table methods.*

1. Introduction

The explosive growth of the big data in recent years has brought great challenges to the scalable similarity search. Among the vast solutions, hash-based approximate nearest neighbor (ANN) search has achieved attractive performance in many applications like visual search [11, 15, 33, 38], object detection [4], visual classification [25], recommendation [20] and image matching [1]. Locality-Sensitive Hashing (LSH) [3, 14] pioneered the hashing research by indexing similar data using similar hash codes, and achieves large-scale search in a constant or sub-linear time. However, since the hash functions are randomly generated in LSH, it usually needs long hash codes to reach a satisfactory performance. As indicated by the literature, the capability of capturing the data structure has significant affects on the performance of many tasks like retrieval and classification [8, 12, 19, 23, 28, 35]. Therefore, many traditional hashing methods attempt to learn data-dependent hash functions that preserve data neighbor structures, and pursue compact, yet informative binary codes by utilizing the complementarity among hash functions [9, 13, 17, 26, 29, 30, 34, 36, 40].

Though the compact codes can achieve compressed storage and efficient computation, they usually fail to satisfy the practical requirement for a desired number of retrieved nearest neighbors. To address this problem, multi-table methods have been studied to build several tables that maximally cover the nearest neighbors by leveraging the table complementarity [1, 4, 7, 11, 21, 24, 27, 37]. As the most widely-used strategy, LSH-based multi-table can faithfully balance recall and precision performance [1, 24, 31], working like multi-index hashing [27]. However, it often requires a huge number of tables without eliminating the table redundance. [37] proposed a sequential learning method to build complementary hash tables, and obtained promising performance with much less tables. [21] further studied the general multi-table construction strategy using bit selection over existing hashing algorithms.

Despite the aforementioned progress, most of these related works still suffer from the lack of table complementarity for the maximum coverage of nearest neighbors when using multiple tables. Moreover, their hash tables are usually learned only from single type of data source. In real-life applications many objects have a set of diverse and complementary descriptors in the form of multiple views (*e.g.*, images can be described by different visual descriptors like SIFT and GIST) [5], and existing hashing research have proved that adaptively combining them can help learn more informative hash functions [16, 22, 32].

In this paper, we aim to learn complementary hash ta-

bles that adaptively incorporates information from different views. For each hash table, by exploiting the exemplar's sensitivity to the neighbor structure in each view, we adaptively fuse multiple features based on exemplars to capture the meaningful nearest neighbors along the manifold. The fused nonlinear feature mapping faithfully helps approximate the inherent data similarities with a low-rank structure, achieving fast computation for hash function learning and out-of-sample extension. Furthermore, we propose a novel exemplar reweighting scheme that sequentially learns complementary tables by eliminating table redundancy in a boosting manner. Such scheme amplifies importance of exemplars shared by previous mis-separated neighbors, and subsequently enlarges their similarities for new table learning without destroying the low-rank properties.

To our best knowledge, this is the first work that learns multiple hash tables from multiple views for nearest neighbor search, which simultaneously and adaptively exploits multi-view information and table correlations. Owing to exemplar based feature fusion and its reweighing scheme, the proposed method is powerful to mutually capture the underlying neighbor structures using multi-view complementary tables, and computationally feasible for large-scale learning and fast online search. Empirical study on several large benchmarks highlights the benefits of our method, with significant performance gains over the several state-of-the-art hashing algorithms and multi-table methods.

The remaining sections are organized as follows. Section 2 presents the multi-view hash function learning for single hash table using the exemplar based feature fusion. In Section 3 we provide the exemplar reweighing scheme for the sequential complementary multi-table construction. Comprehensive experiments on several large datasets are presented in Section 4, followed by conclusions in Section 5.

2. Multi-View Hash Table

In this section, we first define notations that will be used throughout this paper. Assume we are given a set of N training examples $\{\mathbf{x}_i\}_{i=1}^N$ with M views. The data matrix in m-th view can be represented as $\mathbf{X}^{(m)} = [\mathbf{x}_1^{(m)}, \ldots, \mathbf{x}_N^{(m)}] \in \mathbb{R}^{d_m \times N}$, where d_m is the feature dimension in this view.

Our goal is to build L multi-view complementary hash tables $\{\mathcal{T}_l\}_{l=1}^L$, with B hash functions $\mathcal{H} = \{h_j(\cdot)\}_{j=1}^B$ learnt for each table $\mathcal{T}_l$, where $h_j(\cdot) : \mathbb{R}^d \to \{-1, 1\}$ is a binary mapping.

To achieve this goal, next we will first present how to learn single hash table from multiple views, and then study the multiple complementary tables in next section.

2.1. Exemplars based Feature Fusion

In nearest neighbor search, it is rather critical to capture the neighborhood structure underlying the data. Motivated

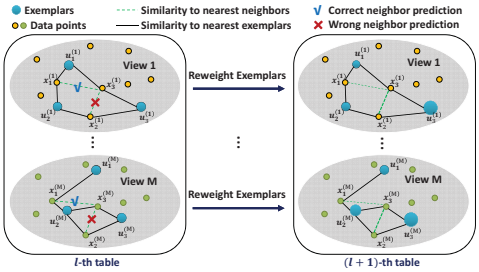


Figure 1. Demonstration of the proposed multi-view complementary hash table method in the sequential learning framework.

by prior exemplar based approximation techniques [18], to address this problem, we adopt a small set of K exemplars (*e.g.*, cluster centers) to adequately cover the data space, and approximately measure the similarities among all N data points with respect to them (See Figure 1). Formally, for N data points in m-th view, K ($K \ll N$) exemplar points $\mathcal{U}^{(m)} = \{\mathbf{u}_k^{(m)} \in \mathbb{R}^{d_m}\}_{k=1}^K$ are employed to characterize the inherent neighbor structures among the feature space. Then each $\mathbf{x}_i^{(m)}$ in m-th view can be distinctively described by its nearest exemplars, forming a new feature representation $\mathbf{z}_i^{(m)}$, with its k-th element $[\mathbf{z}_i^{(m)}]_k$ determined based on the similarity between $\mathbf{x}_i^{(m)}$ and k-th exemplar

$$\left[\mathbf{z}_i^{(m)}\right]_k = \frac{\delta_k^{(m)} \mathcal{K}(\mathbf{x}_i^{(m)}, \mathbf{u}_k^{(m)})}{\sum_{k'=1}^K \delta_k^{(m)} \mathcal{K}(\mathbf{x}_i^{(m)}, \mathbf{u}_{k'}^{(m)})}$$

where $\delta_k^{(m)} \in \{0, 1\}$: $\delta_k^{(m)} = 1$ if and only if exemplar $\mathbf{u}_k^{(m)}$ is one of $\mathbf{x}_i$'s s-nearest ($s \ll K$) exemplars in $\mathcal{U}^{(m)}$ according to the specified kernel function $\mathcal{K}(\cdot, \cdot)$, *e.g.*, Gaussian kernel as the typical one.

With the nonlinear transformation in each view, we adaptively fuse the multiple views of each data point by linearly weighting and concatenating its M feature vectors as one

$$\mathbf{z}_i^* = \frac{1}{\lambda}[\mu_1^r \mathbf{z}_i^{(1)}; \ldots; \mu_M^r \mathbf{z}_i^{(M)}], \tag{1}$$

where $\lambda = \sum_{m=1}^M \mu_m^r$ ($r > 1$) is a normalizer and the notation ";" serves as a column-wise vector concatenation operator. The discriminative weight vector $\boldsymbol{\mu} = [\mu_1, \ldots, \mu_M]^{\mathrm{T}}$, satisfying $\mu_m > 0$ and $\sum_{m=1}^M \mu_m = 1$, balances the importance of each view and can be adaptively learnt.

Since each $\mathbf{z}^{(m)}$ is highly sparse with s nonzero elements, the fused feature representation $\mathbf{z}^*$ contains only sM nonzero entries summing to 1. $\mathbf{z}^*$ intrinsically provides a powerful descriptor that approximate the data similarity incorporating multiple views. Denoting $\mathbf{Z}^* = [\mathbf{z}_1^*, \ldots, \mathbf{z}_N^*]$ and $\boldsymbol{\Lambda} = \mathrm{diag}(\mathbf{Z}^{*\mathrm{T}}\mathbf{1})$, the similarities between N samples can be defined by the following low-rank matrix:

$$\mathbf{S} = \mathbf{Z}^* \boldsymbol{\Lambda}^{-1} \mathbf{Z}^{*\mathrm{T}}, \tag{2}$$

This can be further decomposed as

$$\mathbf{S} = \frac{1}{\lambda} \sum_{m=1}^M \mu_m^r \hat{\mathbf{S}}^{(m)}, \tag{3}$$

Algorithm 1 Learning Single Multi-View Hash Table.

1: **Input**: feature transformation $\mathbf{Z}^{(m)}$ and the similarity matrix $\hat{\mathbf{S}}^{(m)}$ of $\{\mathbf{x}_i\}_{i=1}^N$, $m = 1, \ldots, M$;
2: **Initialize**: the feature weights $\boldsymbol{\mu} = \frac{1}{M}\mathbf{1}$;
3: **repeat**
4: Update the embedding $\mathbf{Y}$ and the hashing projection vectors $\mathbf{W}$ according to (6);
5: Update the feature weights $\boldsymbol{\mu}$ according to (8);
6: **until** Converge
7: Compute the hashing projection vectors $\mathbf{W}^*$ and quantize $\mathbf{Y}$ into binary codes $\mathbf{Y}^*$ by solving (9);
8: **Output**: the hashing projection vectors $\mathbf{W}^*$ and the feature weights $\boldsymbol{\mu}$.

where each component $\hat{\mathbf{S}}^{(m)}$ actually is the similarity matrix $\hat{\mathbf{S}}^{(m)} = \mathbf{Z}^{(m)}\boldsymbol{\Lambda}^{(m)-1}\mathbf{Z}^{(m)\mathrm{T}}$ defined by the transformation $\mathbf{Z}^{(m)} = [\mathbf{z}_1^{(m)}, \ldots, \mathbf{z}_N^{(m)}]$ in m-th view, with $\boldsymbol{\Lambda}^{(m)} = \mathrm{diag}(\mathbf{Z}^{(m)\mathrm{T}}\mathbf{1})$. This fact indicates that the similarity induced from our exemplar based feature fusion is equivalent to the linear combination of corresponding approximated similarities in each view.

2.2. Hash Function Learning

Based on the similarity matrix $\mathbf{S}$ we can learn B hash functions $\mathcal{H} = \{h_j(\cdot)\}_{j=1}^B$ from multiple views for single hash table. Specifically, if we have the hash code matrix $\mathbf{Y}^* = [\mathbf{y}_1^*, \ldots, \mathbf{y}_N^*]^\mathrm{T} \in \{-1, 1\}^{N\times B}$, with each $\mathbf{y}_i^* = [h_1(\mathbf{x}_i), \ldots, h_B(\mathbf{x}_i)]^\mathrm{T}$ as the generated hash code by $\mathcal{H}$ for $\mathbf{x}_i$, the hash codes should minimize the following objective satisfying balanced and uncorrelated constraints [19, 35]

$$\begin{array}{cc} \min_{\boldsymbol{\mu},\mathbf{Y}^*} & \frac{1}{2}\sum_{i=1}^N\sum_{j=1}^N \mathbf{S}_{ij}\|\mathbf{y}_i^* - \mathbf{y}_j^*\|^2 \\ \text{s.t.} & \boldsymbol{\mu} > 0,\ \boldsymbol{\mu}^\mathrm{T}\mathbf{1} = 1 \\ & \mathbf{1}^\mathrm{T}\mathbf{Y}^* = 0,\ \mathbf{Y}^{*\mathrm{T}}\mathbf{Y}^* = N\mathbf{I}_{B\times B}. \end{array} \tag{4}$$

The problem is a NP-hard with the discrete constraint on $\mathbf{Y}^*$. Fortunately, we can approximately solve it by first relaxing binary codes $\mathbf{Y}^*$ to real-valued embedding $\mathbf{Y} \in R^{N\times B}$, and then alteratively optimizing $\mathbf{Y}$ and $\boldsymbol{\mu}$. Next, we list the main steps of the alternating optimization for hash function learning of single multi-view table, given the similarities among data.

$\mathbf{Y}$-Update: When fixing the feature weights $\boldsymbol{\mu}$, the normalization in $\mathbf{Z}^*$ and $\mathbf{S}$ by λ will have no effect on the optimization of projection $\mathbf{Y}$. Without loss of generality we omit it here. Problem 4 turns to a spectral decomposition problem of a Laplacian matrix $\mathbf{L} = \lambda\mathbf{I}_{B\times B} - \mathbf{S}$ [35]. Therefore, the optimal $\mathbf{Y}$ should be the B eigenvectors of $\mathbf{L}$ with the smallest eigenvalues except 0, which are also eigenvectors of $\mathbf{S}$ associated with the B largest eigenvalues (ignoring λ):

$$\begin{array}{cc} \max_{\mathbf{Y}} & \mathrm{tr}(\mathbf{Y}^\mathrm{T}\mathbf{S}\mathbf{Y}) \\ \text{s.t.} & \mathbf{1}^\mathrm{T}\mathbf{Y} = 0,\ \mathbf{Y}^\mathrm{T}\mathbf{Y} = N\mathbf{I}_{B\times B}. \end{array} \tag{5}$$

Directly optimizing the objective is considerably time-consuming ($O(N^3)$), since the decomposition of $\mathbf{S}$ cannot be easily scaled up for large training sets. Fortunately, as indicated by (2), $\mathbf{S}$ is sparse and low-rank, owing to our exemplar based feature fusion. Therefore, the B desired eigenvectors $\mathbf{v}_b$, $b = 1, \ldots, B$ can be easily obtained by solving a much smaller matrix $\boldsymbol{\Lambda}^{-1/2}\mathbf{Z}^{*\mathrm{T}}\mathbf{Z}^*\boldsymbol{\Lambda}^{-1/2}$, whose corresponding eigenvalues σ_b satisfy $\lambda > \sigma_1 \geq \ldots \geq \sigma_B > 0$.

Denoting $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_B]$ and $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1, \ldots, \sigma_B)$, we get the desired spectral embedding $\mathbf{Y}$ for Problem 5

$$\mathbf{Y} = \sqrt{N}\mathbf{Z}^*\boldsymbol{\Lambda}^{-1/2}\mathbf{V}\boldsymbol{\Sigma}^{-1/2} = \mathbf{Z}^*\mathbf{W}, \tag{6}$$

where $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_B]$ with $\mathbf{w}_b = \sqrt{N/\sigma_b}\boldsymbol{\Lambda}^{-1/2}\mathbf{v}_b$.

μ-Update: With $\mathbf{Y}$ fixed, Problem 4 turns to the following problem with respect to $\boldsymbol{\mu}$:

$$\begin{array}{cc} \min_{\boldsymbol{\mu}} & \sum_{m=1}^M \mu_m^r \mathrm{tr}(\mathbf{Y}^\mathrm{T}(\mathbf{I}_{B\times B} - \hat{\mathbf{S}}^{(m)})\mathbf{Y}) \\ \text{s.t.} & \boldsymbol{\mu} > 0,\ \boldsymbol{\mu}^\mathrm{T}\mathbf{1} = 1 \end{array} \tag{7}$$

whose optimal solution is

$$\mu_m = \frac{\mathrm{tr}(\mathbf{Y}^\mathrm{T}(\mathbf{I}_{B\times B} - \hat{\mathbf{S}}^{(m)})\mathbf{Y})^{1/(1-r)}}{\sum_{m=1}^M \mathrm{tr}(\mathbf{Y}^\mathrm{T}(\mathbf{I}_{B\times B} - \hat{\mathbf{S}}^{(m)})\mathbf{Y})^{1/(1-r)}}. \tag{8}$$

Here r adaptively balances the effect of multi-view feature fusion. The strong complementarity among views indicates a large r, which leads to close μ_m in the solution.

By alternatively repeating the above two update steps, we can converge to the optimal $\mathbf{Y}$ and $\boldsymbol{\mu}$ very fast.

$\mathbf{Y}^*$-Update: Finally, the binary codes $\mathbf{Y}^*$ can be efficiently generated by quantizing $\mathbf{Y}$ into $\mathrm{sgn}(\mathbf{Y})$. However, this may bring large quantization loss. For better and balanced coding, we employ the following iterative quantization [9]:

$$\begin{array}{cc} \min_{\mathbf{Y}^*,\mathbf{R}} & \|\mathbf{Y}^* - \mathbf{Y}\mathbf{R}\|_F^2 \\ \text{s.t.} & \mathbf{Y}^* \in \{-1, 1\}^{N\times B}, \mathbf{R}^\mathrm{T}\mathbf{R} = \mathbf{I}_{B\times B}. \end{array} \tag{9}$$

Subsequently, the hashing projection vectors turns to $\mathbf{W}^* = \mathbf{W}\mathbf{R}$ and the binary codes will be $\mathbf{Y}^* = \mathrm{sgn}(\mathbf{Z}^*\mathbf{W}^*)$.

As shown in Algorithm 1, the single multi-view table learning can be interpreted as first nonlinearly transforming features of each sample in multiple views to the exemplar-based feature representation $\mathbf{z}^*$, second linearly projecting $\mathbf{z}^*$ to $\mathbf{y}$, and finally quantizing $\mathbf{y}$ into binary codes $\mathbf{y}^*$.

Remark: The spectral embedding loss objective function in 4 has shown promising power for discriminative binary codes learning capturing neighbor relations [19, 35]. Since directly optimizing the objective is considerably time-consuming, exemplar based approximation has been proposed as a successful technique to scale up the learning in the literature [18]. Along this direction, anchor graph based hashing [19], as the most related work, show great capability of large-scale compact code learning, and achieved encouraging performance for large-scale problems. However,

for multiple hash tables, it is still far beyond both the adaptive information incorporation from multiple views and the strong guarantee of table complementarity, which has been addressed simultaneously by this work.

2.3. Out-of-Sample Extension

In Algorithm 1, the binary codes for the training data can be learned at the training stage. Nevertheless, the explicit hash functions $\mathcal{H} = \{h_j(\cdot)\}_{j=1}^{B}$ are still required for efficient out-of-sample extension in real-world applications. The fact that nearest neighbors usually share similar hash codes motivates us to estimate the code of any new sample based on their neighbors in the training set, and fortunately, using the following theorem we can guarantee that the multi-view hashing process is explicitly equivalent to linear hash functions, which can be directly and efficiently applied to any new sample.

Theorem 1 *Given K exemplar points $\mathcal{U}^{(m)} = \{\mathbf{u}_k^{(m)}\}_{k=1}^{K}$ and $\mu_m \geq 0$ for m-th view, $m = 1, \ldots, M$, define a feature map $z : \{\mathbb{R}^{d_1}, \ldots, \mathbb{R}^{d_M}\} \rightarrow \mathbb{R}^{KM}$ with $r > 1$ for any sample $\mathbf{x}$ as follows*

$$z(\mathbf{x}) = [\mu_1^r \mathbf{z}^{(1)}(\mathbf{x}), \ldots, \mu_M^r \mathbf{z}^{(M)}(\mathbf{x})]^{\mathrm{T}}$$

with $z^{(m)}(\mathbf{x}) = \frac{[\delta_1^{(m)} \mathcal{K}(\mathbf{x}^{(m)}, \mathbf{u}_1^{(m)}), \ldots, \delta_K^{(m)} \mathcal{K}(\mathbf{x}^{(m)}, \mathbf{u}_K^{(m)})]}{\sum_{k=1}^{K} \delta_k^{(m)} \mathcal{K}(\mathbf{x}^{(m)}, \mathbf{u}_k^{(m)})}$. Using the Nyström extension, the binary codes for $\mathbf{x}$ will be

$$\mathbf{y}^* = sgn(\mathbf{W}^{*\mathrm{T}} z(\mathbf{x})).$$

The proof can be completed using the exemplar based feature fusion (please see the supplementary material). The theorem states that for a new sample its hash code can be generated simply by a composite operation: nonlinear feature map $z(\mathbf{x})$, linear projection using $\mathbf{W}^*$ and sgn binarization, which implies explicit hash functions $\mathcal{H}$: $h_j(\mathbf{x}) = \mathrm{sgn}(\mathbf{w}_j^{*\mathrm{T}} z(\mathbf{x}))$, with $\mathbf{w}_j^*$ as the j-th column of $\mathbf{W}^*$.

Now, we can summarize that by using the exemplar based feature fusion the proposed multi-view hash table can be learnt explicitly and efficiently at the offline training stage, and promises fast out-of-sample extension for any new sample at the online searching stage.

3. Complementary Multi-View Tables

In practice, multiple tables complementary to each other together can considerably cover more nearest neighbors, and thus be able to largely boost the overall search performance [10, 37]. In the above section, we have presented how to efficiently learn hash functions for single multi-view hash table, which depends on a low-rank similarity matrix induced from the exemplar based feature fusion. However, as to building multiple complementary hash tables, two critical problems are still left unsolved, *i.e.*, the table complementarity and the low-rank similarity. In this section, we introduce our exemplar reweighting method that simultaneously addresses both problems by sequentially updating the low-rank similarities in a complementary manner.

3.1. Table Complementarity

Motivated by the powerful ensemble learning [6], to build complementary tables, our straightforward solution is sequentially learning each hash table that can correct the prediction errors of previous tables one by one.

We first define an overall Hamming distance $\mathbf{D}_{ij}^{l}$ between any points $\mathbf{x}_i$ and $\mathbf{x}_j$ over l hash tables as the global criteria. Let $\mathbf{D}_{ij}^{0} = 0$, then

$$\mathbf{D}_{ij}^{l} = \min_{k=1,\ldots,l} \sum_{h \in \mathcal{T}_k} \|h(\mathbf{x}_i) - h(\mathbf{x}_j)\|^2. \tag{10}$$

The distance definition should be consistent with the true similarities among data, *i.e.*, it will be small for the true neighbors, and large for others.

Based on $\mathbf{D}$ we can predict the neighbor relations $\mathbf{P}$ of l hash tables: $\mathbf{P}_{ij} = d_e - \mathbf{D}_{ij}^{l}$ on the neighbor pair $\mathbf{x}_i$ and $\mathbf{x}_j$, where d_e is an empirical Hamming radius. A larger $\mathbf{P}_{ij}$ indicates a true neighbor pair prediction between $\mathbf{x}_i$ and $\mathbf{x}_j$.

In order to make the new hash table complementary to previous ones, for each view the similarities on the inconsistent neighbor pairs will be amplified to incur greater concentration at next round. Formally, at the $(l+1)$-th round the similarity matrix $\hat{\mathbf{S}}^{(m)}$ of m-th view will be updated in a boosting-like manner:

$$\hat{\mathbf{S}}_{ij}^{(m)} = \hat{\mathbf{S}}_{ij}^{(m)} \exp(-\alpha^{(m)} \mathbf{P}_{ij}). \tag{11}$$

Here $\alpha^{(m)} = \ln \frac{\sum_{ij} \mathbb{I}(\mathbf{P}_{ij} \hat{\mathbf{S}}_{ij}^{(m)} < 0)}{\sum_{ij} \mathbb{I}(\mathbf{P}_{ij} \hat{\mathbf{S}}_{ij}^{(m)} > 0)}$ with $\mathbb{I}(\cdot)$ as a indicator function. $\alpha^{(m)}$ measures the overall prediction error of previous l tables in m-th view. The similarity updating guides to build a hash table that corrects those previous mistakes over the true nearest neighbors.

Although the sequential similarity update promises the table complementarity, but it will lose the low-rank property of the similarity matrix, and thus degenerates the efficiency of hash function learning and out-of-sample extension for each table using Algorithm 1. Next, we will give our exemplar reweighting scheme that addresses the low-rank similarity updating problem by fully utilizing the exemplars' sensitivity to the neighbor structure.

3.2. Exemplar Reweighting

Since exemplars in each view have great power to characterize the neighbor relations among data, we further employ them to adjust their roles in the similarity approximation. Figure 1 illustrates the intuition of the exemplar reweighing scheme, where if the true neighbors are wrongly predicted (indicated by the red cross) by the l-th table,

then the importance of the exemplars shared by these samples should be increased (displayed by the circle size) to induce a larger similarity between them for the $(l+1)$-th table learning.

Following this intuition, for m-th view we subsequently calibrate the importance of each exemplar in the nonlinear feature representation using weights $\boldsymbol{\pi}^{(m)} = [\pi_1^{(m)}, \ldots, \pi_K^{(m)}]^{\mathrm{T}}$. Therefore, with $\boldsymbol{\Pi}^{(m)} = \mathrm{diag}(\boldsymbol{\pi}^{(m)})$ the reweighted representation turns to

$$\hat{\mathbf{Z}}^{(m)} = \mathbf{Z}^{(m)}\boldsymbol{\Pi}^{(m)}, \tag{12}$$

which induces a new approximated similarity matrix $\mathbf{Z}^{(m)}\boldsymbol{\Lambda}^{(m)-1}\boldsymbol{\Pi}^{(m)}\mathbf{Z}^{(m)\mathrm{T}}$. To preserve the table complementarity, the optimal $\boldsymbol{\pi}^{(m)}$ should force this similarity matrix to be consistent with the desired one in (11):

$$\min_{\boldsymbol{\pi}^{(m)}} \quad \|\mathbf{Z}^{(m)}\boldsymbol{\Lambda}^{(m)-1}\boldsymbol{\Pi}^{(m)}\mathbf{Z}^{(m)\mathrm{T}} - \hat{\mathbf{S}}^{(m)}\|_F^2. \tag{13}$$

The above formulation is equivalent to a least square problem, whose solution can be efficiently obtained.

We normalize the updated feature transformation $\hat{\mathbf{Z}}^{(m)}$

$$\hat{\mathbf{Z}}^{(m)} = \boldsymbol{\Gamma}^{-1}\mathbf{Z}^{(m)}\boldsymbol{\Pi}^{(m)}, \tag{14}$$

where $\boldsymbol{\Gamma} = \mathrm{diag}(\mathbf{Z}^{(m)}\boldsymbol{\Pi}^{(m)}\mathbf{1})$. The following theoretical result further guarantees that with the updated similarity based on exemplar reweighting in each view, we can sequentially learn hash functions for the multi-view complementary hash tables using Algorithm 1:

Proposition 1 *Theorem 1 still holds when using the nonlinear feature map based on exemplar reweighing:*

$$\mathbf{z}^{(m)}(\mathbf{x}) = \frac{[\delta_1^{(m)}\pi_1^{(m)}\mathcal{K}(\mathbf{x}^{(m)}, \mathbf{u}_1^{(m)}), \ldots, \delta_K^{(m)}\pi_K^{(m)}\mathcal{K}(\mathbf{x}^{(m)}, \mathbf{u}_K^{(m)})]}{\sum_{k=1}^{K}\delta_k^{(m)}\pi_k^{(m)}\mathcal{K}(\mathbf{x}^{(m)}, \mathbf{u}_k^{(m)})}.$$

Algorithm 2 summarizes our approach to constructing L multi-view complementary hash tables $\{\mathcal{T}_l\}_{l=1}^{L}$, which enjoys efficient multiple table learning and fast online search. Please see the computational complexity analysis in the supplementary material.

4. Experiments

In this section we will comprehensively evaluate the proposed method named Multi-View Complementary Hash tables (MVCH for short). Since there is no related work regarding hash tables with multiple views in the literature, in this paper we compare it to several naive solutions: state-of-the-art well-known unsupervised hashing methods: *Local Sensitive Hashing* (LSH) [3], *Iterative Quatization* (ITQ) [9] and *Anchor Graph Hashing* (AGH) [19], multiple feature hashing methods: *Multiple View Hashing* (MVH) [16], *Multiple Feature Hashing* (MFH) [32] and *Unsupervised Multiple Feature Kernel Hashing* (UMFKH) [22], and multiple table construction methods *Complementary Hashing* (CH) [37] and *Bit Selection* (BS) [21].

Algorithm 2 Learning Multi-View Complementary Hash Tables (MVCH).

Input: the training set $\{\mathbf{x}_i\}_{i=1}^{N}$ with M views, the exemplars $\mathcal{U}^{(m)}$, $m = 1, \ldots, M$;
Initialize: the feature representations $\mathbf{Z}^{(m)}$ by (1), the similarity matrix $\hat{\mathbf{S}}^{(m)}$ by (1), and $\mathbf{D}_{ij}^{0} = 0$;
for l = 1 to L **do**
  Compute the exemplar weights $\boldsymbol{\pi}^{(m)}$ for each view by solving (13);
  Update feature representations $\hat{\mathbf{Z}}^{(m)}$ by (14);
  Optimize projection vectors $\mathbf{W}^{*}$ and feature weights $\boldsymbol{\mu}$ using Algorithm 1 with input $\hat{\mathbf{Z}}^{(m)}$ and $\hat{\mathbf{S}}^{(m)}$;
  Build hash table $\mathcal{T}_l$ using $\mathbf{W}^{*}$ and the nonlinear feature map $z(\cdot)$ in (10) and (15);
  Update $\mathbf{D}_{ij}^{l}$ and $\hat{\mathbf{S}}_{ij}^{(m)}$ by (10) and (11);
end for
Output: multi-view complementary tables $\{\mathcal{T}_l\}_{l=1}^{L}$.

Among the baselines, only CH and BS can exploit multiple complementary hash tables, guaranteeing that the true nearest neighbors missed from one hash table are more likely to be found in other tables, and meanwhile only MVH, MFH and UMFKH can adaptively incorporate information from multiple features. Since baselines like LSH, AGH, MVH, MFH and UMFKH initially aim to learn compact hash codes, to build multiple tables respectively using these methods we generate a desired number of hash functions and evenly partition them into different tables following *Multi-Index Hashing* [27]. For LSH, AGH, CH and BS that cannot originally support multiple views, we simply concatenate the features of different views as one.

4.1. Datasets and Protocols

Multiple hash tables have been widely applied to nearest neighbor search in many areas including image search [11,37], object detection [4] and image matching [1]. Without loss of generality, we evaluate our method on image search, where three popular large image datasets: **CIFAR-10** (60K), **TRECVID** (260K) and **NUS-WIDE** (270K) are adopted in our experiments. For simplicity and similar to prior multiple feature work [16,32], we adopt different types of visual features for each set as the distinct views to verify the efficiency and effectiveness of our proposed method.

CIFAR-10 contains 60K 32×32 color images of 10 classes and 6K images in each class. For each image, we extract 300-D bag-of-words (BoW) quantized from dense SIFT features and 384-D GIST feature.

TRECVID [39] is a large-scale image dataset built from the TRECVID 2011 Semantic Indexing annotation set with 126 fully labeled concepts, from which we select 25 most-frequent concepts. For each image, we extract 512-D GIST feature and 1000-D spatial pyramid bag-of-words feature.

Table 1. Hash table lookup performance of different multi-table methods on CIFAR-10, TRECVID and NUS-WIDE.

	Methods	RH2				PH2			
		$L=1$	$L=4$	$L=8$	$L=16$	$L=1$	$L=4$	$L=8$	$L=16$
CIFAR-10	LSH	0.71 $\pm$0.11	2.67 $\pm$0.16	5.06 $\pm$0.48	9.07 $\pm$0.25	14.67 $\pm$0.51	15.25 $\pm$0.27	15.01 $\pm$0.35	14.71 $\pm$0.19
	AGH	3.15 $\pm$0.17	4.53 $\pm$0.17	5.63 $\pm$0.16	7.59 $\pm$0.18	20.10 $\pm$0.32	18.28 $\pm$0.14	16.66 $\pm$0.13	14.97 $\pm$0.11
	ITQ	3.49 $\pm$0.19	5.86 $\pm$0.28	7.68 $\pm$0.20	10.81 $\pm$0.15	19.60 $\pm$0.38	16.98 $\pm$0.33	15.92 $\pm$0.28	15.04 $\pm$0.19
	MFH*	4.68 $\pm$0.49	-	-	-	22.67 $\pm$0.12	-	-	-
	MVH	1.58 $\pm$0.03	1.43 $\pm$0.02	2.31 $\pm$0.02	4.03 $\pm$0.02	20.54 $\pm$0.07	15.08 $\pm$0.08	12.94 $\pm$0.04	11.72 $\pm$0.03
	UMFKH	1.10 $\pm$0.07	1.98 $\pm$0.18	3.02 $\pm$0.07	4.50 $\pm$0.04	19.24 $\pm$0.70	15.53 $\pm$0.44	14.15 $\pm$0.17	12.76 $\pm$0.07
	CH	0.71 $\pm$0.03	3.46 $\pm$0.15	5.66 $\pm$0.16	8.73 $\pm$0.33	20.06 $\pm$0.38	19.33 $\pm$0.41	19.06 $\pm$0.37	18.47 $\pm$0.40
	BS	3.05 $\pm$0.55	5.80 $\pm$0.56	7.90 $\pm$0.57	10.68 $\pm$0.59	16.18 $\pm$0.26	15.31 $\pm$0.34	15.01 $\pm$0.32	14.44 $\pm$0.23
	MVCH	**5.55** $\pm$0.20	**9.18** $\pm$0.28	**11.19** $\pm$0.31	**13.32** $\pm$0.39	**26.21** $\pm$0.78	**25.02** $\pm$0.75	**24.38** $\pm$0.69	**23.76** $\pm$0.59
TRECVID	LSH	1.23 $\pm$0.38	4.78 $\pm$0.52	7.64 $\pm$0.40	12.25 $\pm$0.19	22.93 $\pm$0.92	22.67 $\pm$0.93	22.36 $\pm$0.88	22.06 $\pm$0.75
	AGH	4.75 $\pm$0.64	5.67 $\pm$0.70	6.15 $\pm$0.66	7.09 $\pm$0.59	22.97 $\pm$0.42	22.70 $\pm$0.44	22.39 $\pm$0.47	21.77 $\pm$0.48
	ITQ	4.26 $\pm$0.23	8.02 $\pm$0.61	10.90 $\pm$0.69	11.66 $\pm$0.77	23.99 $\pm$0.44	22.89 $\pm$0.55	22.22 $\pm$0.36	21.71 $\pm$0.31
	MFH	1.96 $\pm$0.23	3.73 $\pm$0.26	6.57 $\pm$0.52	9.93 $\pm$0.66	23.11 $\pm$0.27	20.75 $\pm$0.34	19.19 $\pm$0.35	18.42 $\pm$0.40
	MVH	0.73 $\pm$0.03	0.55 $\pm$0.03	0.82 $\pm$0.03	1.36 $\pm$0.03	22.77 $\pm$0.79	21.16 $\pm$0.62	19.47 $\pm$0.49	18.39 $\pm$0.43
	UMFKH	0.36 $\pm$0.03	0.99 $\pm$0.07	1.61 $\pm$0.12	2.00 $\pm$0.06	23.81 $\pm$0.52	22.23 $\pm$0.24	21.46 $\pm$0.19	19.44 $\pm$0.13
	CH	0.43 $\pm$0.01	3.61 $\pm$0.44	6.10 $\pm$0.45	8.92 $\pm$0.61	24.13 $\pm$0.83	22.89 $\pm$0.26	22.14 $\pm$0.26	21.45 $\pm$0.39
	BS	2.66 $\pm$0.39	5.80 $\pm$0.64	8.87 $\pm$1.26	**13.02** $\pm$0.93	23.65 $\pm$0.33	22.75 $\pm$0.28	22.40 $\pm$0.30	21.99 $\pm$0.30
	MVCH	**5.07** $\pm$1.20	**9.09** $\pm$0.77	**10.94** $\pm$0.86	**12.77** $\pm$0.94	**24.46** $\pm$0.62	**23.86** $\pm$0.56	**23.59** $\pm$0.57	**23.34** $\pm$0.58
NUS-WIDE	LSH	0.38 $\pm$0.07	1.29 $\pm$0.08	2.82 $\pm$0.26	5.46 $\pm$0.29	30.32 $\pm$1.01	30.23 $\pm$1.15	30.15 $\pm$1.53	29.71 $\pm$1.52
	AGH	1.31 $\pm$0.04	1.79 $\pm$0.07	2.12 $\pm$0.07	2.82 $\pm$0.09	34.34 $\pm$2.21	32.27 $\pm$2.14	30.62 $\pm$1.89	28.82 $\pm$1.65
	ITQ	2.01 $\pm$0.30	**4.51** $\pm$0.68	**6.25** $\pm$0.27	8.11 $\pm$0.43	33.36 $\pm$1.36	30.05 $\pm$1.00	28.63 $\pm$1.65	28.46 $\pm$1.29
	MFH	0.32 $\pm$0.01	0.57 $\pm$0.01	0.87 $\pm$0.01	1.44 $\pm$0.01	34.50 $\pm$0.79	30.55 $\pm$0.53	28.14 $\pm$0.48	26.51 $\pm$0.45
	MVH	0.15 $\pm$0.01	0.32 $\pm$0.01	0.60 $\pm$0.01	1.15 $\pm$0.01	32.63 $\pm$1.08	26.82 $\pm$0.67	25.36 $\pm$0.58	24.57 $\pm$0.53
	UMFKH	0.31 $\pm$0.01	0.94 $\pm$0.18	1.35 $\pm$0.17	1.63 $\pm$0.04	**36.46** $\pm$0.73	29.71 $\pm$0.28	28.03 $\pm$0.40	26.26 $\pm$0.44
	CH	0.22 $\pm$0.00	3.14 $\pm$0.03	5.18 $\pm$0.21	7.33 $\pm$0.22	36.23 $\pm$1.63	32.96 $\pm$1.52	33.12 $\pm$1.28	32.69 $\pm$1.43
	BS	0.77 $\pm$0.02	1.84 $\pm$0.01	3.01 $\pm$0.04	5.44 $\pm$0.18	33.17 $\pm$1.75	30.25 $\pm$1.02	29.48 $\pm$0.84	29.30 $\pm$0.87
	MVCH	**2.17** $\pm$0.17	4.18 $\pm$1.03	**6.02** $\pm$1.07	**8.95** $\pm$1.15	35.67 $\pm$1.06	**34.88** $\pm$1.03	**34.60** $\pm$1.45	**34.29** $\pm$1.59

* We didn't get a reasonable performance by tuning parameters of MFH on CIFAR-10 when using multiple hash tables. Similar results are shown in Figure 2.

NUS-WIDE [2] comprises over 269,000 images with 81 ground truth concept tags, of which we only consider 25 most frequent tags ('sky', 'animal', etc.). Besides, multiple visual features have been provided already in this set, and we select three presentative features: 128-D wavelet texture, 225-D block-wise color moments and 500-D SIFT-based BoW histograms.

For each dataset, we construct a training and a testing set respectively with 5,000 and 3,000 random samples. The groundtruth for each testing query is defined as those samples in the database with at least one common label as the query. Since we manually selected complementary features, a large r can achieve satisfying performance, and we empirically set $r = 12$ in all experiments. For AGH and MVCH using exemplar based feature transformation, we employ 300 exemplars generated by k-means clustering.

There are two common hashing search schemes in the literature: (1) Hamming distance ranking: all points in the database are ranked according to the Hamming distances from the query, and then the top ranked samples are returned as the retrieved results. (2) Hash table lookup: a lookup table is constructed using the binary codes, and points falling within certain Hamming radius from the query codes are returned as retrieved results. To comprehensively evaluate multiple table construction methods, we extend the two search schemes for multiple table search. The former ranks all candidate points according to their Hamming distances over all tables to the query defined in (10), while the later retrieves points indexed in the buckets of all tables within certain Hamming radius from the query code. Usually a small searching radius (we use 2, *i.e.*, $d_e = 2$) is used to avoid the expensive computation stemming from combinatorial explosion [19]. To suppress the randomness, all experimental results are reported by averaging over 10 runs.

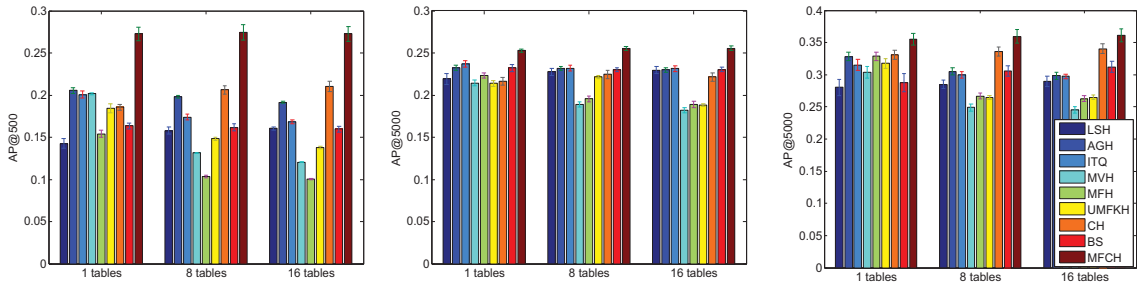
4.2. Multi-Table Search Evaluation

Instead of simply concatenating multiple features as one in traditional hashing algorithms, multiple feature hashing algorithms like MVH, MFH and UMFKH consider the complementary relations between different views when learning hash functions. However, for multiple tables they still fail to exploit the mutual benefit between tables. The proposed multi-table construction method adaptively combines features of multiple views using the exemplar based feature fusion, and meanwhile avoids the table redundance by sequentially reweighting anchors in each feature space.

To verify this point, we respectively build a different number of hash tables using different methods. Prior research indicates that for a balanced search performance the optimal number of hash functions in each table should be close to $\log_2 N$ for N points [27, 31]. Therefore, for CIFAR-10, TRECVID and NUS-WIDE, we respectively choose 16, 18 and 18 hash functions for each hash table. Next, with this setting we will comprehensively evaluate these construction methods in terms of both hash table lookup and Hamming distance ranking.

4.2.1 Hash Table Lookup

Building several hash tables can activate more buckets from multiple tables in the searching process, and thus locate more true nearest neighbors, which subsequently improves the recall performance. However, in traditional multi-table methods, it usually cannot guarantee (even sharply reduces) the search accuracy due to a large portion of false positive and redundant results.



(a) AP @500 on CIFAR-10 (b) AP @5000 on TRECVID (c) AP @5000 on NUS-WIDE

Figure 2. Average precision using Hamming distance ranking on CIFAR-10, TRECVID and NUS-WIDE.

Table 1 lists the hash table lookup results, *i.e.*, precision and recall within Hamming radius 2 (PH2 and RH2 for short), comparing the proposed method MVCH with other baselines on the three datasets. It can be observed that the recall performance of AGH using 1-16 hash tables increases from 3.15% to 7.59%, but the precision decreases from 20.10% to 14.97%. A similar phenomenon is also observed when using MVH. This is because that these native solutions ignore the effect of redundancy between tables. Instead, CH, BS and our MVCH relieve the degeneration by simultaneously learning informative tables and enhancing their complementarity.

On CIFAR-10, though the precision of MVCH, as well as LSH and CH, reduces slightly when using more hash tables, but by comparing their recall we can find that CH and MVCH can significantly boost the performance when using more hash tables, which indicates that both methods eliminating table redundance are able to build hash tables that widely cover more true neighbor pairs. Compared with existing multi-table methods CH and BS over LSH that equally treat each type of feature, our MVCH can further exploit discriminative information from multiple sources by adaptively determining the importance of each view, and thus achieves much higher performance than CH and BS in terms of both recall and precision, up to *e.g.*, 52.58% recall and 28.64% precision gains using 16 hash tables.

The results on TRECVID and NUS-WIDE further verify the fact that though building multiple tables can improve the recall performance, it usually sharply reduces the search accuracy (especially using CH, MFH and MVH) due to the irrelevant and redundant points contained in the results. This is also true for most baselines, mainly due to that these native solutions do not eliminate the effect of table redundancy. On the contrast, our MVCH gets higher precision and recall rate by relieving such degeneration using multiple complementary hash tables. Moreover, MVCH further exploits the complementarity between different views to discover the inherent neighbor structure, and thus outperforms CH and BS consistently using a different number of tables.

Figure 4 reports the training and search time per query using different methods on CIFAR-10. MVCH spends a little more time than others on offline training (but less than BS and close to CH). However, since in practice usually only tens of hash tables are required, it is quite beneficial that MVCH can get significant performance gains without costing much more training time. Besides, MVCH can achieve fast online search using almost the same time as baselines including linear, nonlinear and multi-view hashing.

4.2.2 Hamming Distance Ranking

Besides hash table lookup, Hamming distance ranking is another widely employed search technique without explicitly building multiple tables, owing to its fast implementation using binary operations.

Figure 2 plots the average precision (AP) at certain cutting point of the returning list using Hamming distance ranking on three datasets. From the figure, we can easily observe that our MVCH, simultaneously incorporating both multiple features and the coupled hash tables, consistently achieves the best performance over different types of baselines in all cases. For instance, on CIFAR-10, TRECVID and NUS-WIDE MVCH using 16 tables, it gets 24.33%, 11.19% and 24.67% precision gains over the best competitor CH or BS among all baselines. Moreover, the proposed method slightly increases its overall performance in terms of both recall and precision when using more hash tables, while most of the others decrease. More experimental results can be found in the supplemental material.

4.3. The Exemplar Effect

The proposed method builds complementary hash tables utilizing the fact that the exemplars show sensitivity to the neighbor structures: (1) the exemplar based feature fusion, adaptively incorporating information from multiple views, helps efficiently learn discriminative hash functions for each table; (2) the calibrated exemplars accurately capture the similarity updates and thus induce the complementarity between tables. Besides the above comparison with naive solutions, next we will study the effect of the two characteristics of our method by varying the algorithm settings.

Table 2. Hash table lookup performance of the proposed MVCH with different settings on CIFAR-10.

SETTINGS	RH2				PH2			
	$L=1$	$L=4$	$L=8$	$L=16$	$L=1$	$L=4$	$L=8$	$L=16$
F1	$2.96_{\pm 0.09}$	$4.16_{\pm 0.10}$	$5.16_{\pm 0.12}$	$6.97_{\pm 0.13}$	$16.62_{\pm 0.24}$	$15.52_{\pm 0.15}$	$14.42_{\pm 0.10}$	$13.28_{\pm 0.08}$
F1+ER	$2.96_{\pm 0.09}$	$3.44_{\pm 0.12}$	$3.92_{\pm 0.13}$	$5.05_{\pm 0.19}$	$16.62_{\pm 0.24}$	$16.50_{\pm 0.23}$	$16.40_{\pm 0.24}$	$16.17_{\pm 0.22}$
F2	$3.29_{\pm 0.23}$	$4.75_{\pm 0.25}$	$5.83_{\pm 0.23}$	$7.75_{\pm 0.23}$	$24.99_{\pm 0.58}$	$22.11_{\pm 0.54}$	$19.39_{\pm 0.39}$	$16.71_{\pm 0.25}$
F2+ER	$3.29_{\pm 0.23}$	$4.41_{\pm 0.20}$	$5.32_{\pm 0.35}$	$6.46_{\pm 0.35}$	$24.99_{\pm 0.58}$	$24.53_{\pm 0.48}$	$24.14_{\pm 0.38}$	$23.64_{\pm 0.35}$
F1F2	$3.12_{\pm 0.17}$	$4.40_{\pm 0.21}$	$5.42_{\pm 0.21}$	$7.28_{\pm 0.21}$	$19.94_{\pm 0.20}$	$18.04_{\pm 0.16}$	$16.31_{\pm 0.17}$	$14.62_{\pm 0.14}$
F1+F2+ER	**5.55** $_{\pm 0.20}$	**9.18** $_{\pm 0.28}$	**11.19** $_{\pm 0.31}$	**13.32** $_{\pm 0.39}$	**26.21** $_{\pm 0.78}$	**25.02** $_{\pm 0.75}$	**24.38** $_{\pm 0.69}$	**23.76** $_{\pm 0.59}$

4.3.1 On Feature Fusion

To illustrate the benefits from exemplar based feature fusion, on CIFAR-10 we compare our method with different settings (but all with exemplar reweighting, ER for short): using exemplar based fusion of two views ("F1+F2+ER", *i.e.*, the standard MVCH in Sec. 4.2), and using single view: BoW ("F1+ER") or GIST ("F2+ER"). Table 2 reports PH2 and RH2 of these methods using hash table lookup. It can be observed that GIST feature shows more promising performance than BoW feature, and meanwhile simply concatenating multiple features as one ("F1F2") hardly captures the most useful information from multiple sources (See results using one table where ER hasn't been activated). Figure 3 shows the precision using Hamming distance ranking of different methods, where we can obtain the similar observation that feature fusion without considering their correlations even performs worse than single view. However, in both evaluation our method achieves the best performance with significant gains in all cases. This indicates that MVCH can faithfully exploit the meaningful neighbor structures along manifolds of multiple views.

4.3.2 On Table Complementarity

To further investigate the exemplars' effect on table complementarity, we respectively compare the performance of MVCH with the following settings: using single view without ER (*i.e.*, "F1" and "F2") and with ER (*i.e.*, "F1+ER" and "F2+ER"), using two views fused based on direct concatenation ("F1F2") and that based on exemplars ("F1+F2+ER"). Note that when only building one table from single view without ER, MVCH can be regarded degenerated to AGH [19]. By comparing the performance of these methods listed in Table 2, we can observe that building more hash tables increases the recall performance of "F1" and "F2", but sharply lowers their precisions. Figure 3 further proves our observation that those methods without suppressing the redundance among tables decrease their precision when using more hash tables. Instead, the proposed exemplar reweighting strategy in "F1+ER" and "F2+ER" enables their multiple tables together to cover more true neighbors, and thus significantly boosts the search performance balancing both precision and recall. Moreover, both Table 2 and Figure 3 demonstrate that our MVCH can build complementary tables and achieves the best performance by simultaneously integrating both adaptive multiple feature fusion and exemplar reweighting.

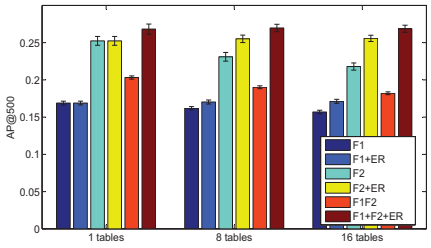


Figure 3. Hamming distance ranking performance of MVCH using different settings on CIFAR-10.

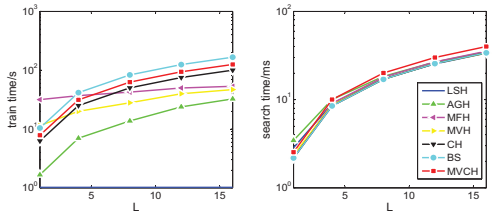


Figure 4. Training and search time (per query) of different multi-table methods on CIFAR-10.

5. Conclusion

We first presented a complementary hash table construction method that learns hash tables by adaptively incorporating information from multiple views. The exemplar based feature fusion was first introduced to capture the inherent neighbor structures among data by linearly combining the nonlinear feature transformation in each view. Based on the feature fusion, for single table the proposed method efficiently learns discriminative hash functions with multiple views. For complementary hash tables, a boosting manner with an exemplar reweighting scheme is applied to reducing the table redundance. Besides the discriminative power of multiple tables, the proposed method also enjoys efficient learning and fast code generation for unseen samples. Comprehensive evaluation on large-scale benchmarks demonstrates its good practicability and performance. Future work can further concentrate on the theoretical analysis of the table complementarity and optimal parameters.

Acknowledgment

This work was partially supported by the National Natural Science Foundation of China (61402026, 61370125 and 61572388), the Foundation of State Key Lab of Software Development Environment (2014ZX-07 and 2015ZX-04), the Key Science and Technology Program of Shaanxi Province (2014K05-16), and the Innovation Foundation of BUAA for PhD Graduates.

References

[1] J. Cheng, C. Leng, J. Wu, H. Cui, and H. Lu. Fast and accurate image matching with cascade hashing for 3d reconstruction. In *IEEE CVPR*, pages 4321–4328, 2014.

[2] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng. Nus-wide: a real-world web image database from national university of singapore. In *ACM CIVR*, 2009.

[3] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *SCG*, pages 253–262, 2004.

[4] T. Dean, M. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, and J. Yagnik. Fast, accurate detection of 100,000 object classes on a single machine. In *IEEE CVPR*, pages 1–8, 2013.

[5] C. Deng, R. Ji, W. Liu, X. Gao, and D. Tao. Visual reranking through weakly supervised multi-graph learning. 2013.

[6] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28:2000, 1998.

[7] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *VLDB*, pages 518–529, 1999.

[8] Y. Gong, S. Kumar, H. A. Rowley, and S. Lazebnik. Learning binary codes for high-dimensional data using bilinear projections. In *IEEE CVPR*, 2013.

[9] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *IEEE CVPR*, pages 817–824, june 2011.

[10] J. He, S.-F. Chang, R. Radhakrishnan, and C. Bauer. Compact hashing with joint optimization of search accuracy and time. In *IEEE CVPR*, pages 753–760, 2011.

[11] J. He, J. Feng, X. Liu, T. Cheng, T.-H. Lin, H. Chung, and S.-F. Chang. Mobile product search with bag of hash bits and boundary reranking. In *IEEE CVPR*, pages 3005–3012, 2012.

[12] K. He, F. Wen, and J. Sun. K-means hashing: An affinity-preserving quantization method for learning binary compact codes. In *IEEE CVPR*, pages 2938–2945, June 2013.

[13] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon. Spherical hashing. In *IEEE CVPR*, pages 2957–2964, 2012.

[14] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *ACM STOC*, pages 604–613, 1998.

[15] B. Kulis and T. Darrell. Learning to hash with binary reconstructive embeddings. In *NIPS*, pages 1–8, 2009.

[16] S. Kumar and R. Udupa. Learning hash functions for cross-view similarity search. In *IJCAI*, pages 1360–1365, 2011.

[17] X. Li, G. Lin, C. Shen, A. van den Hengel, and A. Dick. Learning hash functions using column generation. In *ICML*, 2013.

[18] W. Liu, J. He, and S.-F. Chang. Large graph construction for scalable semi-supervised learning. In *ICML*, pages 679–686, 2010.

[19] W. Liu, J. Wang, S. Kumar, and S.-F. Chang. Hashing with graphs. In *ICML*, pages 1–8, 2011.

[20] X. Liu, J. He, C. Deng, and B. Lang. Collaborative hashin. In *IEEE CVPR*, June 2014.

[21] X. Liu, J. He, and B. Lang. Reciprocal Hash Tables for Nearest Neighbor Search. In *AAAI*, pages 626–632, Bellevue, Washington, 2013.

[22] X. Liu, J. He, and B. Lang. Multiple feature kernel hashing for large-scale visual search. *Pattern Recognition*, 47(2):748 – 757, 2014.

[23] X. Liu, J. He, B. Lang, and S.-F. Chang. Hash bit selection: a unified solution for selection problems in hashing. In *IEEE CVPR*, 2013.

[24] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li. Multi-probe lsh: efficient indexing for high-dimensional similarity search. In *VLDB*, pages 950–961, 2007.

[25] Y. Mu, G. Hua, W. Fan, and S.-F. Chang. Hash-svm: Scalable kernel machines for large-scale visual classification. In *IEEE CVPR*, June 2014.

[26] M. Norouzi and D. J. Fleet. Cartesian k-means. In *IEEE CVPR*, pages 2938–2945, June 2013.

[27] M. Norouzi, A. Punjani, and D. J. Fleet. Fast search in hamming space with multi-index hashing. In *IEEE CVPR*, pages 3108–3115, 2012.

[28] F. Shen, W. Liu, S. Zhang, Y. Yang, and H. T. Shen. Learning binary codes for maximum inner product search. In *IEEE ICCV*, 2015.

[29] F. Shen, C. Shen, W. Liu, and H. Tao Shen. Supervised discrete hashing. In *IEEE CVPR*, pages 37–45, June 2015.

[30] F. Shen, C. Shen, Q. Shi, A. van den Hengel, and Z. Tang. Inductive hashing on manifolds. In *IEEE CVPR*, Oregon, USA, 2013.

[31] M. Slaney, Y. Lifshits, and J. He. Optimal parameters for locality-sensitive hashing. *Proceedings of the IEEE*, 100(9):634–640, 2012.

[32] J. Song, Y. Yang, Z. Huang, H. T. Shen, and R. Hong. Multiple feature hashing for real-time large scale near-duplicate video retrieval. In *ACM MM*, pages 423–432, 2011.

[33] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for scalable image retrieval. In *IEEE CVPR*, pages 3424–3431, 2010.

[34] J. Wang, W. Liu, A. Sun, and Y.-G. Jiang. Learning hash codes with listwise supervision. In *IEEE ICCV*, pages 3032–3039, 2013.

[35] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, pages 1–8, 2008.

[36] Y. Xia, K. He, P. Kohli, and J. Sun. Sparse projections for high-dimensional binary codes. In *IEEE CVPR*, 2015.

[37] H. Xu, J. Wang, Z. Li, G. Zeng, S. Li, and N. Yu. Complementary hashing for approximate nearest neighbor search. In *IEEE ICCV*, pages 1631–1638, 2011.

[38] G. Ye, D. Liu, J. Wang, and S.-F. Chang. Large-scale video hashing via structure learning. In *IEEE ICCV*, 2013.

[39] F. Yu, R. Ji, M.-H. Tsai, G. Ye, and S.-F. Chang. Weak attributes for large-scale image retrieval. In *IEEE CVPR*, pages 2949–2956, 2012.

[40] X. Yu, S. Kumar, Y. Gong, and S. Chang. Circulant binary embedding. In *ICML*, 2014.