Joint Versus Independent Multiview Hashing for Cross-View Retrieval

Peng Hu, Xi Peng, Member, IEEE, Hongyuan Zhu, Member, IEEE, Jie Lin, Member, IEEE, Liangli Zhen, and Dezhong Peng

Abstract—Thanks to the low storage cost and high query speed, cross-view hashing (CVH) has been successfully used for similarity search in multimedia retrieval. However, most existing CVH methods use all views to learn a common Hamming space, thus making it difficult to handle the data with increasing views or a large number of views. To overcome these difficulties, we propose a decoupled CVH network (DCHN) approach which consists of a semantic hashing autoencoder module (SHAM) and multiple multiview hashing networks (MHNs). To be specific, SHAM adopts a hashing encoder and decoder to learn a discriminative Hamming space using either a few labels or the number of classes, that is, the so-called flexible inputs. After that, MHN independently projects all samples into the discriminative Hamming space that is treated as an alternative ground truth. In brief, the Hamming space is learned from the semantic space induced from the flexible inputs, which is further used to guide view-specific hashing in an independent fashion. Thanks to such an independent/decoupled paradigm, our method could enjoy high computational efficiency and the capacity of handling the increasing number of views by only using a few labels or the number of classes. For a newly coming view, we only need to add a view-specific network into our model and avoid retraining the entire model using the new and previous views. Extensive experiments are carried out on five widely used multiview databases compared with 15 state-of-the-art approaches. The results show that the proposed independent hashing paradigm is superior to the common joint ones while enjoying high efficiency and the capacity of handling newly coming views.

Index Terms—Common hamming space, cross-view retrieval, decoupled cross-view hashing network (DCHN), multiview hashing, multiview representation learning.

I. INTRODUCTION

With the rapid growth of multiview data, such as image, text, and video on the Internet, there are increasing demands on developing cross-view methods for a variety of applications [1]–[6]. Among them, cross-view retrieval has arisen great interest from the community, which aims to retrieve the interested content across different views/modalities, for example, retrieving the corresponding text counterpart for a given image query. Due to the low storage cost and high query speed of hash codes [7], [8], cross-view hashing (CVH) has achieved promising performance and is becoming increasingly popular for the large-scale multimedia retrieval. Although CVH has been paid more attention to by both academia and industry [9], [10], there still remain many challenges. Especially, different views may lie in completely disparate spaces with large semantic gaps, thus resulting in inferior retrieval performance.

To eliminate the semantic gap, numerous CVH methods have been proposed to project multiview data into a common Hamming space by narrowing the heterogeneous gap. In general, most existing multiview hashing approaches could be roughly classified into two categories, that is: 1) shallow [11]–[13] and 2) deep methods [9], [10], [14]. The shallow approaches usually learn some single-layer linear or nonlinear transformations to project multiview data into a shared Hamming space [15], [16]. One major limitation of linear methods is that they may be incapable of capturing the high-level nonlinear semantics of real-world data. To address this limitation, some kernel methods [12], [17] have been proposed. However, it is still an open issue and a daunting task to choose a suitable kernel function [18]. To adaptively capture the nonlinearity in data, several recent works attempted to use the deep neural network (DNN) to learn a common hash space across different views in an unsupervised [19] or supervised [9], [10], [20] way.

To be specific, although the aforementioned CVH methods have achieved promising performance, they need all views to jointly learn the common Hamming space as shown in Fig. 1(a), thus facing the following two disadvantages.
II. RELATED WORK

Multiview learning has been paid more and more attention from academic and industry communities for multimedia retrieval [24]–[26]; multiview clustering [27], [28]; disease analysis [29]; etc. Furthermore, it is very interesting that multisource learning could benefit from multiview learning [30], which will extend the application scope of multiview learning. Hashing has been widely used in many applications due to its great advantages, for example, low storage cost and high accuracy. However, hashing has traditionally only been applied to a limited number of categories, whereas in real-world problems, the number of categories is typically large. To handle large-scale views, various hashing models based on the traditional single-view hashing paradigm have been developed. One of the most widely used paradigms is the joint representation learning paradigm. However, this paradigm is not sufficient for dealing with multiple views, and it cannot be extended to deal with a large number of views. To solve these problems, we propose a novel multiview hashing method, called decoupled CVH networks (DCHN), which consists of a SHAM and multiple multiview hashing networks (MHNs). In brief, SHAM encodes the flexible input into the discriminative Hamming space in which the corresponding hash code is further decoded to reconstruct the input as shown in Fig. 2. After that, the learned Hamming space is treated as an alternative ground truth, which guides the optimization of MHN (see Fig. 1). The major difference from the existing hashing paradigm is that we do not jointly learn a common Hamming space from all views, but rather, we independently train each view-specific network with its own independent objective function $L_k$, thus overcoming the above limitations suffered from the common paradigms.

The advantages of our DCHN are two-fold. First, thanks to our “hashing from labels” paradigm, all view-specific MHNs and the corresponding loss functions are independent of each other. Therefore, these networks can be independently trained regardless of when and where they should be trained, that is, our method breaks the temporal and spatial connections of view-specific networks. For a new coming view, existing CVH methods have to retrain their models using all views (old and new views) due to their joint hashing paradigms, whereas our method will only establish and optimize a new MHN, thanks to our independent hashing paradigm. Second, our DCHN even allows being trained in a sequential manner, which is helpful to the scenario with limited resources, that is, we would perform training and inference view-by-view. These advantages are brought by the designed independent training strategy regardless of any specific form of the deep networks. As a result, our method could run on the low resource devices (e.g., mobile devices) even though a large number of views and newly coming views are available.

The main contributions of our work could be summarized as follows.

1) To the best of our knowledge, the proposed SHAM could be the first multiview hashing method that learns from a few labels or only the number of categories, that is, the so-called flexible inputs. Such an advantage significantly makes our method highly attractive in practice. For example, limited by privacy and copyright, multiple owners of different views cannot share their data with each other. In such a case, it is highly expected to develop a method like our DCHN, which could first use some insensitive data (e.g., the class number) to learn the Hamming space and then use the Hamming space to separately learn the hash code and perform retrieval.

2) Thanks to the proposed paradigm of “hashing from labels,” a novel deep multiview hashing method (DCHN) is proposed, which could independently learn the hash code for different views, thus embracing the capacity of handling increasing views and large-scale views. To the best of our knowledge, such a hashing method with the aforementioned capacities has been less touched in the previous studies.
query speed, including multiview hashing [22], [25]; image retrieval [31]–[33]; etc. In this work, we only focus on the multiview hashing learning that could be roughly classified into shallow and deep models.

The key to these methods is to learn a common space shared by different views. To be specific, traditional CVH methods [12], [21], [22] learn two transformations to project the cross-view data into a common Hamming space shared across different views. For example, Lin et al. [21] proposed a supervised CVH method, called semantics-preserving hashing (SePH), using the semantic affinity of different views. Moreover, Lin et al. extended SePH by learning the predictive models (e.g., linear ridge regression, logistic regression, or kernel logistic regression) as the hashing functions in each view to project the corresponding view into the common Hamming space [12], such as SePH with logistic regression (SePH LR). Li et al. [22] presented a supervised linear subspace ranking hashing framework (LSRH) to project two views into a shared Hamming space, which employs the Hamming distance to measure the similarity between different views. Ding et al. [34] developed a so-called rank-order preserving hashing method (RoPH) for a cross-view similarity search by introducing a novel regression-based rank-order preserving loss.

Recently, DNN has been successfully applied to numerous multiview problems for learning a common space [9], [10], [35]–[37]. These methods adopt different technologies to learn a common Hamming space and achieve promising results, for example, pairwise constraints [35]; deep quantization [36]; adversarial learning [20], [37]; etc. Specifically, Jiang and Li [9] proposed deep cross-modal hashing (DCMH) by integrating feature learning and hash-code learning into a unified deep framework. Li et al. [20] proposed a self-supervised adversarial hashing method (SSAH), which aims to utilize the ability of the adversarial learning to model the multiview data distribution. Deng et al. [10] proposed a triplet-based deep hashing method (TDH) for the large-scale cross-view retrieval by using a deep CNN to perform feature learning and hashing in an end-to-end manner. Hu et al. [38] utilized the variational inference and the similarity relationship of samples to project the samples from different views into a single shared Hamming space.

Although some works [21], [39] and our DCHN are two-step approaches, they are remarkably different in given aspects. First, the previous two-step methods need the labels of all views to learn hash codes in the first step, whereas our method could only use a few labels or the class number to learn the semantic hash encoder and decoder. Second, although some one-step methods (e.g., SSAH [20]) could learn from labels like our DCHN, they should jointly use the label information and views to learn the hash codes as shown in Fig. 1(a). In contrast, our method could only use the class number or a few available labels to learn the semantic hash encoder and decoder. Therefore, different from the pioneer works, including SSAH [20], the hashing learning process of our method is decoupled rather than joint learning, thus being capable of handling large-scale and increasing views. In conclusion, different from the aforementioned traditional and deep cross-view methods, our method does not utilize all views to jointly learn a common Hamming representation. Instead, we utilize very little input information (a few labels or only the number of classes) to learn neural networks for our SHAM and then use the label-induced Hamming space to optimize the view-specific MHN. As a result, our method is more efficient and effective because it could handle an increasing number of views while using a few computational resources.

III. PROPOSED METHOD

As shown in Fig. 3, our DCHN employs a SHAM and v MHNs to learn the unified hash codes for all views.

A. Problem Formulation

For ease of presentation, some definitions are given below which will be used in the remaining sections. Let the kth view be denoted by $x^{k} = \{x^{k}_i\}_{i=1}^{N_k}$, where $x^{k}_i$ is the ith sample of the kth view, and $N_k$ is the number of the samples from the kth view. Besides, let $y^{k} = \{y^{k}_l\}_{l=1}^{c}$ be the label set, where $y^{k}_l \in \mathbb{R}^{c \times 1}$ is a binary-value label vector for the sample $x^{k}_i$ and $c$ is the number of categories. If the ith sample of the kth view belongs to the jth class, $y^{k}_{ij} = 1$; otherwise, $y^{k}_{ij} = 0$. For the single-label data, their semantic labels only contain one nonzero value. In contrast, the labels will be with multiple nonzero elements for multilabel data. In the experiments, we will show the effectiveness of our method to these two cases. However, in the following, we will not specifically discuss this issue for clarity.

Multiview hashing aims to learn a common hash space $\mathcal{B} = \{B^k\}_{k=1}^{K}$ shared by multiple views, where $v$ is the number of views, $B^k = \{b^k_1, \ldots, b^k_i, \ldots, b^k_{N_k}\}$ is the discrete code matrix of the kth view, $b^k_j \in \{-1, 1\}^L$ is the binary code of $x^k_j$, and $L$ is the length of the hash code. In the Hamming space, the similarity between different points is measured by the Hamming distance. As the Hamming distance $H(b^k_i, b^k_j)$ and the inner product $(1/2)(b^k_i \cdot b^k_j)$ are related by $H(b^k_i, b^k_j) = (1/2)(L - (b^k_i \cdot b^k_j))$, we therefore use the inner product $(1/2)(b^k_i \cdot b^k_j)$ to measure the similarity $\Gamma^k_{ij}$ of two binary codes $(b^k_i$ and $b^k_j)$, that is, $\Gamma^k_{ij} = (1/2)(b^k_i \cdot b^k_j)$. With the above notations, our MHN aims to learn v view-specific hashing functions $f_{k}(-)$ to project the corresponding view into the Hamming space obtained by our SHAM.

B. Framework

As shown in Fig. 3, the proposed DCHN adopts our SHAM (see Fig. 2) consisting of the encoder $\mathbf{W}$ and the decoder $\mathbf{W}^T$ and v MHNs to learn the unified hash codes for all views. In this section, we elaborate on the implementation details of SHAM and the network architectures of MHN.
In this section, we first introduce the formulation of our SHAM and then present the details of the optimization procedure. As shown in Fig. 2, SHAM is a traditional autoencoder that is with only one hidden layer shared by the encoder and decoder. The encoder aims to project the flexible input into a discriminative Hamming space in which the obtained hash codes are further used to reconstruct the input. Formally

$$\arg \min_{W,V,Q} \|S - VWS\|_F^2 \quad \text{s.t.} \quad WS = Q, \quad Q \in \{-1, 1\}^{L \times m} \quad (1)$$

where $\| \cdot \|_F$ is the Frobenius norm, $W$ and $V$ are the hidden layer transformations of the encoder and decoder, $Q$ is the semantic binary matrix, $m$ is the number of objects in the semantic space, and $S \in \mathbb{R}^{c \times m}$ is the semantic input matrix induced by the flexible inputs. In brief, when $m$ labels are available, denoted as DCHN$_m$. $S$ is a label assignment matrix constructed by the corresponding $m$ label vectors, that is, $S \in \mathbb{R}^{c \times m}$. However, there are not any available labels to construct $S$ when only the class number $c$ is available, denoted as DCHN$_0$. Motivated by the widely used one-hot label encoding [40], which could maximize the difference between distinct classes, $S$ could be constructed according to, for example, a label-indicator matrix of which each column is a different one-hot class vector when only the class number $c$ is available, that is, $S \in \mathbb{R}^{c \times c}$.

Fig. 3. Framework of the proposed DCHN method. $g$ is the output of the corresponding view (i.e., image, text, video, etc.). $o$ is the semantic hash code that is computed by the corresponding label $y$ and semantic hashing transformation $W$. $W$ is computed by the proposed semantic hashing autoencoder module (SHAM). $\text{sgn}$ is an elementwise sign function. $L_R$ and $L_H$ are hash reconstruction [see (12)] and semantic hashing [see (11)] functions, respectively. In the training stage, first, $W$ is used to recast the label $y$ as a ground-truth hash code $o$. Then, the obtained hash code is used to guide view-specific networks with a semantic hashing reconstruction regularizer. Such a learning scheme makes the training stage, first, $W$ is computed by the proposed semantic hashing autoencoder module

$$\text{arg min}_{W,V} \|S - VWS\|_F^2 \quad \text{s.t.} \quad WS = Q, \quad Q \in \{-1, 1\}^{L \times m}. \quad (2)$$

It is difficult to solve an objective function with the hard constraint such as $WS = Q$ [42]. Therefore, to optimize the objective in (2), we relax the constraint into a soft constraint and rewrite the objective as

$$\arg \min_{W,Q} \|S - W^TQ\|_F^2 + \lambda \|WS - Q\|_F^2 \quad \text{s.t.} \quad Q \in \{-1, 1\}^{L \times m} \quad (3)$$

where $\lambda > 0$ is a balance parameter. It is well known that (3) is intractable as $Q$ are with binary values. Different from most of the previous CVH methods which relaxes the above discrete problem as a continuous one, our SHAM enforces the discrete constraint $Q \in \{-1, 1\}^{L \times m}$ to directly learn the hash codes $Q$. Equation (3) could be solved using a tractable alternating minimization algorithm as follows.

First, fixing $W$, it gives the derivative of (3) as follows:

$$- W(S - W^TQ) + \lambda (Q - WS) = 0 \quad (4)$$

then

$$Q = \text{sgn}((1 + \lambda)(WW^T + \lambda I)^{-1}WS) \quad (5)$$

To further simplify our SHAM, we adopt the tied weights via $V = W^T$ [41], [42]. Then, (1) could be rewritten as follows:

$$\text{arg min}_{W,Q} \|S - W^TQ\|_F^2 \quad \text{s.t.} \quad WS = Q, \quad Q \in \{-1, 1\}^{L \times m}. \quad (2)$$

It is difficult to solve an objective function with the hard constraint such as $WS = Q$ [42]. Therefore, to optimize the objective in (2), we relax the constraint into a soft constraint and rewrite the objective as
The class number of the hashing transformation $W$ through SHAM, the length of the hash code $L$, the balance parameter $\beta$, the learning rate $\alpha$, and the batch size $N_b$.

Require: The training data set of all views $\{X^k\}_{k=1}^V$, the corresponding label sets $\{Y^k\}_{k=1}^V$, the learned semantic hashing transformation $W$ through SHAM, the length of the hash code $L$, the balance parameter $\beta$, the learning rate $\alpha$, and the batch size $N_b$.

1: parfor $k = 1, 2, \ldots, V$ do
2: Randomly initialize the parameters $\Theta_k$ of the $k$th view-specific network.
3: while not converge do
4: Randomly sample $N_b$ points from the $k$th view $\{X^k, Y^k\}$ to construct a view-specific mini-batch.
5: For each sampled point $x^k_i$, calculate $g^i_k = f_k(x^k_i; \Theta_k)$ by forward-propagation, and the semantic hash code of the label $y^k_i$ according to (9).
6: Compute the MHN loss for the $k$th view-specific neural network with (13) in the mini-batch.
7: Update $\Theta_k$ by minimizing the obtained loss by using back propagation as follows:
   $\Theta_k \leftarrow \Theta_k - \alpha \triangledown_{\Theta_k} L_k.$
8: end while
9: end parfor
10: return Optimized view-specific hashing models.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.
This objective function allows training DCHN using SGD and its variants in an end-to-end manner. The parallel optimization process is summarized in Algorithm 2. With the learned $\Theta^*_k$, the hash code of $x_i$ can be obtained as follows:

$$b^*_k = \text{sgn}(f_k(x_i^*; \Theta_k^*)) \in \{-1, 1\}^L_e. \quad (15)$$

IV. EXPERIMENTAL STUDY

To evaluate the proposed method, we compare our DCHN with 15 state-of-the-art cross-view methods on five datasets in terms of effectiveness and efficiency. The used datasets contain PKU XMedia [44], MIRFLICKR-25K [45], IAPR TC-12 [46], NUS-WIDE [47], and MS-COCO [48]. Moreover, we also conduct an ablation study and parameter sensitivity to investigate the effectiveness of our method.

A. Datasets and Compared Methods

In this section, we briefly introduce the five aforementioned datasets. For a fair comparison, we partition the database into retrieval, training, and query sets by following [20] and [44]. In Table I, we show the general statistics of the five datasets. In our experiments, all tested methods adopt the same features for fair comparisons, that is, the above seven shallow methods and three deep methods (i.e., Doc2Vec, etc.) are not fine tuned in the training process. To the best of our knowledge, NUS-WIDE and MS-COCO datasets. In the experiments, the length of hash codes $L_e$ is set as 16, 32, 64, and 128 bits for a comprehensive evaluation. To probe the performance of a variant of our method, called the “Baseline.” The only one difference between the Baseline and our DCHN is that the former directly adds a softmax classifier layer [52] on the top of MHN with the cross-entropy loss function to investigate the effectiveness of our SHAM.

B. Implementation Details

The proposed DCHN approach is trained on two NVIDIA GTX 1080Ti in PyTorch. We use the ADAM optimizer [53] to train our approach. The batch size and maximal epoch are set as 64 and 100, respectively. The learning rate $\alpha$ is empirically set as 0.0001 for each view. For all the used datasets, the image features are extracted by a trained 19-layer VGGNet [40] model that has been pretrained on ImageNet. Specifically, the used features are output from the fc7 layer of the VGGNet. For the MS-COCO database, we use a trained Doc2Vec model$^1$ [54], which has been pretrained on Wikipedia, to extract 300-D features from the sentences. For the other datasets, the text feature of each document is a bag-of-words vector (BoW). For the other views of PKU XMedia (i.e., audio, 3-D, and video), the features are given by the authors. In our view-specific MHNs, three fully connected layers are utilized to learn the common hash codes for all views. Each FC layer follows a ReLU layer except the last layer. The numbers of hidden units of these FC layers are, respectively, 4096, 4096, and 64, 32, 64, and 128 bits for a comprehensive evaluation. To investigate the ability of our method to

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Class</th>
<th>View</th>
<th>Database</th>
<th>Train</th>
<th>Query</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>PKU XMedia</td>
<td>20</td>
<td>Image, Text</td>
<td>4,000</td>
<td>4,000</td>
<td>1,000</td>
<td>4,096D VGG</td>
</tr>
<tr>
<td>MIRFLICKR-25K</td>
<td>24</td>
<td>Image, Text</td>
<td>20,015</td>
<td>20,015</td>
<td>10,000</td>
<td>4,096D VGG</td>
</tr>
<tr>
<td>IAPR TC-12</td>
<td>255</td>
<td>Image, Text</td>
<td>20,000</td>
<td>20,000</td>
<td>10,000</td>
<td>4,096D VGG</td>
</tr>
<tr>
<td>NUS-WIDE</td>
<td>21</td>
<td>Image, Text</td>
<td>188,321</td>
<td>188,321</td>
<td>10,000</td>
<td>4,096D VGG</td>
</tr>
<tr>
<td>MS-COCO</td>
<td>80</td>
<td>Image, Text</td>
<td>117,218</td>
<td>117,218</td>
<td>10,000</td>
<td>4,096D VGG</td>
</tr>
</tbody>
</table>

$^1$https://github.com/jhlau/doc2vec
learn the common Hamming space with the flexible inputs, two variants of our method (i.e., DCHN0 and DCHN100) are also investigated. For DCHN0, only the number of categories is available in SHAM. Regarding DCHN100, 100 labels are available. From the experimental results, we can draw the following conclusions.

### TABLE III

Performance Comparison in Terms of MAP Scores on the MIRFLICKR-25K and IAPR TC-12 Datasets

<table>
<thead>
<tr>
<th>Method</th>
<th>SHAM</th>
<th>MIRFLICKR-25K</th>
<th>Image → Text</th>
<th>Image → Tag</th>
<th>IAPR TC-12</th>
<th>Image → Text</th>
<th>Image → Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>0.581</td>
<td>0.520</td>
<td>0.533</td>
<td>0.578</td>
<td>0.544</td>
<td>0.556</td>
<td>0.579</td>
</tr>
<tr>
<td>SePh [21]</td>
<td>0.729</td>
<td>0.738</td>
<td>0.744</td>
<td>0.753</td>
<td>0.762</td>
<td>0.764</td>
<td>0.769</td>
</tr>
<tr>
<td>SePh[12]</td>
<td>0.729</td>
<td>0.738</td>
<td>0.744</td>
<td>0.753</td>
<td>0.762</td>
<td>0.764</td>
<td>0.769</td>
</tr>
<tr>
<td>RoPh [34]</td>
<td>0.733</td>
<td>0.744</td>
<td>0.749</td>
<td>0.757</td>
<td>0.759</td>
<td>0.768</td>
<td>0.771</td>
</tr>
<tr>
<td>LRSH [22]</td>
<td>0.756</td>
<td>0.780</td>
<td>0.788</td>
<td>0.772</td>
<td>0.786</td>
<td>0.791</td>
<td>0.802</td>
</tr>
<tr>
<td>KDLFH [23]</td>
<td>0.734</td>
<td>0.755</td>
<td>0.770</td>
<td>0.764</td>
<td>0.780</td>
<td>0.794</td>
<td>0.797</td>
</tr>
<tr>
<td>MTHF [13]</td>
<td>0.581</td>
<td>0.571</td>
<td>0.645</td>
<td>0.584</td>
<td>0.556</td>
<td>0.633</td>
<td>0.531</td>
</tr>
<tr>
<td>DJSRH [14]</td>
<td>0.620</td>
<td>0.630</td>
<td>0.645</td>
<td>0.620</td>
<td>0.626</td>
<td>0.645</td>
<td>0.649</td>
</tr>
<tr>
<td>DCMI [9]</td>
<td>0.737</td>
<td>0.754</td>
<td>0.763</td>
<td>0.731</td>
<td>0.760</td>
<td>0.773</td>
<td>0.770</td>
</tr>
<tr>
<td>SSHA [20]</td>
<td>0.797</td>
<td>0.809</td>
<td>0.810</td>
<td>0.782</td>
<td>0.797</td>
<td>0.799</td>
<td>0.790</td>
</tr>
</tbody>
</table>

### TABLE IV

Performance Comparison in Terms of MAP Scores on the NUS-WIDE and MS-COCO Datasets

<table>
<thead>
<tr>
<th>Method</th>
<th>NUS-WIDE</th>
<th>Image → Text</th>
<th>Image → Tag</th>
<th>MS-COCO</th>
<th>Image → Text</th>
<th>Image → Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>0.281</td>
<td>0.337</td>
<td>0.263</td>
<td>0.341</td>
<td>0.299</td>
<td>0.339</td>
</tr>
<tr>
<td>SePh [21]</td>
<td>0.644</td>
<td>0.652</td>
<td>0.661</td>
<td>0.664</td>
<td>0.654</td>
<td>0.662</td>
</tr>
<tr>
<td>SePh[12]</td>
<td>0.607</td>
<td>0.624</td>
<td>0.646</td>
<td>0.650</td>
<td>0.609</td>
<td>0.646</td>
</tr>
<tr>
<td>RoPh [34]</td>
<td>0.619</td>
<td>0.636</td>
<td>0.652</td>
<td>0.614</td>
<td>0.625</td>
<td>0.649</td>
</tr>
<tr>
<td>LRSH [22]</td>
<td>0.323</td>
<td>0.367</td>
<td>0.364</td>
<td>0.403</td>
<td>0.325</td>
<td>0.365</td>
</tr>
<tr>
<td>KDLFH [23]</td>
<td>0.316</td>
<td>0.367</td>
<td>0.381</td>
<td>0.404</td>
<td>0.319</td>
<td>0.379</td>
</tr>
<tr>
<td>MTHF [13]</td>
<td>0.265</td>
<td>0.435</td>
<td>0.434</td>
<td>0.445</td>
<td>0.243</td>
<td>0.418</td>
</tr>
<tr>
<td>DJSRH [14]</td>
<td>0.433</td>
<td>0.453</td>
<td>0.467</td>
<td>0.442</td>
<td>0.457</td>
<td>0.468</td>
</tr>
<tr>
<td>DCMI [9]</td>
<td>0.569</td>
<td>0.395</td>
<td>0.612</td>
<td>0.621</td>
<td>0.548</td>
<td>0.573</td>
</tr>
<tr>
<td>SSHA [20]</td>
<td>0.636</td>
<td>0.636</td>
<td>0.637</td>
<td>0.610</td>
<td>0.553</td>
<td>0.676</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>NUS-WIDE</th>
<th>Image → Text</th>
<th>Image → Tag</th>
<th>MS-COCO</th>
<th>Image → Text</th>
<th>Image → Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>0.648</td>
<td>0.660</td>
<td>0.669</td>
<td>0.683</td>
<td>0.668</td>
<td>0.687</td>
</tr>
<tr>
<td>SePh [21]</td>
<td>0.652</td>
<td>0.661</td>
<td>0.647</td>
<td>0.684</td>
<td>0.668</td>
<td>0.687</td>
</tr>
<tr>
<td>SePh[12]</td>
<td>0.654</td>
<td>0.671</td>
<td>0.681</td>
<td>0.651</td>
<td>0.668</td>
<td>0.687</td>
</tr>
</tbody>
</table>
1) Deep learning-based methods do not always outperform the traditional methods, and some shallow methods even perform better than the deep methods. The potential reason is that the used deep features contain higher level semantic information, which could boost the performance of the traditional cross-hashing methods.

2) All evaluated methods require using all views to jointly train their models. By incorporating the interview information into the training stage, some approaches outperform our DCHN in a few tasks. However, it should be pointed out that our DCHN can achieve the best performance using only 100 available labels without utilizing the interview information.

3) Directly using labels to decouple the hashing learning (i.e., Baseline) is inferior to other CVH methods, which indicates that the semantic information cannot be directly transferred into a common Hamming space with a separate training manner. Thus, we should elaborately design the decoupling methods for multiview hashing tasks like our DCHN.

4) Most existing CVH methods are specially designed for binary-view cases, which cannot be directly utilized to address the multiview problem (more than two views). In contrast, our DCHN could not only handle multiview data but also the unfixed views, while achieving the best performance even though comparing with real-value multiview methods. Besides the comparisons with the MAP score, we also adopt the precision–recall curves w.r.t. the code length of 128 to evaluate the performance on the MIRFLICKR-25K, IAPR TC-12, NUS-WIDE, and MS-COCO datasets (see Figs. 4 and 5). The result shows that our DCHN could also obtain encouraging results.

Finally, we evaluate the retrieval performance of our methods comparing with the deep CVH methods in terms of qualitative results on the MS-COCO dataset as show in Fig. 6. The correct results are the retrieved samples in the retrieval database that share at least one same class with a given query. From this figure, we can see that our DCHN achieves the best top retrieval results with a few labels (100 labels). In conclusion, our DCHN can embrace more advantages (e.g., more flexible) without losing any performance.

D. Efficiency Analysis About Increasing Views

To investigate the efficiency of our view-independent training paradigm, we report the time costs of three variants of our method on the XMedia database in Table V. To be specific,
“DCHN” means that all the views are jointly used to train the model as the most existing CVH method. “DCHNp” means that all MHNs are trained in parallel at the same time and the reported time is to train all the views. “DCHNs” means that each MHN is separately trained and only one view is trained at the same time. In other words, the time cost of DCHNs could be regarded as the time for handling a newly coming view. For comparisons, we adopt DCMH, SSAH, and DJSRH as the baselines due to two reasons. On one hand, these three methods are based on DNNs and adopt the joint-view hashing paradigm. On the other hand, these methods are with GPU codes, whereas the methods, such as SePH, are shallow methods which are with only CPU codes. In the experiment, the maximal epochs of all the compared methods are set as 20 for a fair comparison within an acceptable time. Note that all tested methods could not converge in the same epochs, and we only focus on evaluating the efficiency of view-independent training on the epoch level. From Section IV-F, we could see that our method could approach convergence near the 20th epoch. Furthermore, our SHAM could precompute $W$ only once on the available labels or class numbers before training MHN. It could converge very fast as shown Fig. 8 and cost little time to compute $W$ (e.g., only 2.14 s for 50 iterations on an Intel i9-10900X CPU@3.70 GHz). Thus, we only compare the efficiency of our MHN with the other methods. Our view-independent training paradigm also could improve the training efficiency on two-view cases, for example, NUS-WIDE, MS-COCO, etc.

From the results, one could have the following conclusions. First, multiview methods (i.e., MAN and our method) are significantly more efficient than the two-view methods (i.e., DCMH, SSAH, and DJSRH) to handle multiple views (more than two views). Second, the proposed parallel method (DCHNp) is remarkably more efficient than the joint learning methods (i.e., MAN, DCMH, SSAH, and DJSRH). Such dominance is more distinct when the view number increases. For example, our method (DCHNs) can only cost much less time and resources to train a new view-specific model for a new view instead of the entire model as other methods. From the table, we can see that our view-independent training paradigm (DCHNp) can remarkably speed up the cross-view training up to 60.73% comparing with DCHN. Furthermore, for a new view, DCHNs can reduce the time up to 96.85% comparing with DCHN and MAN. Note that all these three variants are with the same retrieval accuracy with the same configurations (i.e., random seed, hyperparameters, etc.). Therefore, our DCHN is more efficient to handle new views and increasing views than the existing multiview methods. Such dominance is more outstanding comparing with these two-view methods.

E. Parameter Analysis and Ablation Study

To determine the value of $\lambda$ and $\beta$, we randomly select some samples (2000 for each dataset) from the retrieval database to serve as the validation set by following [20]. In this section, Fig. 7 investigates the influence of these two parameters on NUS-WIDE with the code length of 32, as well as the ablation study. From the result, one could see that the best performance is achieved when $\lambda = 1$ and $\beta = 0.01$. 

---

Fig. 6. Cross-view retrieval result examples on the MS-COCO dataset by our proposed approach as well as compared methods DCMH and SSAH. For a given query (image/text on the left part), the top ten retrieval results (text/image on the right part) are shown on the above. In these examples, the correct retrieval results are with green borders, while the wrong results are with red borders. (a) Ten retrieval result examples for an image query. (b) Ten retrieval result examples for a text query.
Furthermore, the performance will decrease if the Hashing encoder (when $\beta = 1$) or the decoder (when $\beta = 0$) is removed, which indicates that both the encoder and the decoder contribute to the cross-view retrieval performance.

Furthermore, to investigate the impact of the number of available labels, we compare the performance of our DCHN on the different numbers of available labels in terms of MAP scores on MS-COCO with 32 bits. The comparison results are shown in Table VI. From the results, we could see that the real labels could improve the performance of DCHN0 which does not use any labels. More labels could bring better retrieval performance in a certain range (i.e., 0–100 in Table VI). Furthermore, our DCHN could fast achieve a satisfied retrieval accuracy in a certain number of labels, and more labels could not bring significant improvement, thus our DCHN is insensitive to the number of available labels.

### F. Convergence Analysis

We also evaluate the convergence of our method on the MS-COCO dataset. Fig. 8(a) and (b) shows the objective value of (3) versus a different number of iterations with the number of classes and 100 available labels on the MS-COCO dataset, respectively. From Fig. 8(a) and (b), we can see that our SHAM can quickly converge to a certain range. Moreover, Fig. 8(c) shows the losses of MHN versus different numbers of epochs for the image view on the MS-COCO dataset, and Fig. 8(d) shows the losses of MHN versus different numbers of epochs for the text view on the MS-COCO dataset. From the figures, one could see that the proposed MHN converges before the 100th epoch and the changing rates are much faster before the 20th epoch than later epochs. Therefore, we set the maximum epoch as 100. Note that the cross-view retrieval results of the proposed method are reported on the trained models of the last epoch, which is different from other methods that report the best performance throughout their training stages.

### G. Visualization of Learned Representations

To visually investigate the discrimination of common representations learned by different cross-view methods, we adopt the t-SNE approach [55] to embed the samples from the PKU XMedia dataset into a 2-D space as shown in Fig. 9. Note that the most CVH methods cannot simultaneously project multiple views into a common Hamming space. Therefore, we only compare our method with two real-valued methods. From this figure, we can see that the learned representations of these cross-view methods from different views can overlap with each other indicating that they can project different views into a common space. The supervised methods can compact the samples with the same class and scatter the samples from different categories. Although the unsupervised method (MCCA) can project different views into a common space, the samples are scattered without any clustering center in the common space. Therefore, discrimination in multiview data is important for cross-view retrieval. From Fig. 9, we also can see that these methods attempt to project different views into a common space and separate the samples of different classes from each other. The degree of compactness for each class is consistent with the MAP results of cross-view retrieval. Obviously, our DCHN can make the different classes more scattered and the same ones more compact. That is to say, the proposed method can obtain more discriminative information from the cross-view data, which is consistent with the MAP scores for cross-view retrieval tasks.
In this article, we proposed DCHNs to handle the data with an unfixed number of views. The major novelty of our idea is that DCHN does not jointly learn a common Hamming space as existing works did. Instead, we first learned a Hamming space from the flexible input via SHAM and then used it to perform view-specific hashing via the corresponding MHN. Such a hashing paradigm makes separately training view-specific networks possible, thus enjoying the advantages of handling large-scale views and increasing views. Extensive experiments showed that the proposed DCHN achieves state-of-the-art cross-view retrieval performance on three benchmark datasets while enjoying high computational efficiency. As for the future work, we attempt to extend our method to separately learn discrete representations from very few labeled data, which is more difficult in cross-view retrieval.

**REFERENCES**


