Deep Supervised Hashing with Anchor Graph

Yudong Chen[1,2], Zhihui Lai[1,3,*], Yujuan Ding[4], Kaiyi Lin[5], Wai Keung Wong[4]
[1]College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China
[2]Guangdong Key Laboratory of Intelligent Information Processing, Shenzhen University, Shenzhen, China
[3]Shenzhen Institute of Artificial Intelligence and Robotics for Society, Shenzhen, China
[4]Institute of Textiles and Clothing, The Hong Kong Polytechnic University, Hong Kong, China
[5]School of Software and Microelectronics, Peking University, Beijing, China

Abstract

Recently, a series of deep supervised hashing methods were proposed for binary code learning. However, due to the high computation cost and the limited hardware's memory, these methods will first select a subset from the training set, and then form a mini-batch data to update the network in each iteration. Therefore, the remaining labeled data cannot be fully utilized and the model cannot directly obtain the binary codes of the entire training set for retrieval. To address these problems, this paper proposes an interesting regularized deep model to seamlessly integrate the advantages of deep hashing and efficient binary code learning by using the anchor graph. As such, the deep features and label matrix can be jointly used to optimize the binary codes, and the network can obtain more discriminative feedback from the linear combinations of the learned bits. Moreover, we also reveal the algorithm mechanism and its computation essence. Experiments on three large-scale datasets indicate that the proposed method achieves better retrieval performance with less training time compared to previous deep hashing methods.

1. Introduction

Hashing methods are widely used to learn a set of binary codes for feature representation. Since the features only contain values 0 and 1, we can rapidly measure the hamming distances between the data points on large-scale datasets for image retrieval. With the desirable binary property, hashing has been extensively applied in some related areas, such as large-scale clustering [36], collaborative filtering [34] and sketch retrieval [26, 19].

Existing hashing methods can be divided into two categories, i.e., data-independent methods and data-dependent methods. The traditional data-independent method Locality Sensitive Hashing (LSH) [6] can obtain binary codes very fast by using random projection matrix. Due to the simplicity, some extensions of LSH are also developed [4, 14]. However, these methods require more bits to represent the data for high retrieval accuracy.

On the other hand, the data-dependent methods can obtain more effective binary codes by using large number of training data to learn the hashing functions. The unsupervised methods, such as Spectral Hashing (SH) [30] and Locally Linear Hashing (LLH) [10] aim at preserving the manifold structure of the dataset in binary space. Since it is time-consuming to compute the affinity matrix, the Anchor Graph Hashing (AGH) [20] designs an anchor graph to implicitly characterize the local structure. Besides, The Iterative Quantization (ITQ) [7] and Jointly Sparse Hashing (JSH) [15] focus on reducing the information loss by adding a rotation matrix to their optimization models. To utilize the labeled information of training data, Semi-Supervised Hashing (SSH) [28], Latent Factor Hashing (LFH) [35] and Supervised Discrete Hashing (SDH) [25] are proposed for discriminative binary code learning. Inspired by the outstanding performance on single-modal retrieval, some hashing methods, such as Co-Regularized Hashing (CRH) [38], Supervised Matrix Factorization Hashing (SMFH) [27] and Discriminant Cross-modal Hashing (DCMH) [32], are designed for cross-modal retrieval. However, as these methods are based on hand-crafted features, they lack the ability of feature learning.

To tackle this problem, hashing methods based on deep neural network are introduced, which are called deep hashing methods [24, 1]. In recent years, deep neural network has attracted great attention since it can effectively characterize the non-linear property of data and greatly improve the classification performance [9]. To utilize the deep neural network for feature extraction, some deep hashing methods are proposed including the Deep Semantic Ranking Based Hashing (DSRH) [37], Deep Supervised Hashing (DSH) [18] and Convolutional Neural Network Hashing (CNNH)

*Corresponding author: Z. Lai (lai_zhi_hui@163.com).

[31]. Since some of the deep hashing methods use two independent stages to learn the binary codes, the learned deep features may be the sub-optimal solutions for binarization. Recent works showed that the end-to-end hashing methods can greatly improve the quality of the learned binary codes [24, 11, 33]. One of the representative methods is the Deep Pairwise Supervised Hashing (DPSH) [17]. Based on the framework of DPSH, Deep Supervised Hashing with Triplet Labels (DTSH) [29] extends the pairwise labels to triplet labels in the objective function, and Deep Supervised Discrete Hashing (DSDH) [16] adds a discriminant term to update the binary codes to further exploit the labeled information.

The end-to-end deep hashing methods can greatly improve the retrieval performance. However, because of the high computation cost and limited storage space, these methods usually select a subset from the training set to update the network. Therefore, some supervised information of the training data is ignored during the iterative learning procedure and the model cannot directly obtain the optimal binary codes of the training set. As such, the model is less discriminative and the information loss will inevitably increase in the process of binary code learning. Based on these observations, it is desirable to develop a more discriminative and effective method to improve the performance by taking full use of the available labeled information. In this paper, a framework called Deep Anchor Graph Hashing (DAGH) is proposed for deep hashing and efficient binary code learning. The main contributions are listed as follows:

- We propose an effective deep hashing method by regarding the samples in subset as anchors and design a regression formulation to establish the connections between the anchors and all the binary codes. As such, the exact binary codes of training set can be obtained and the problem of information loss can be avoided.
- Through the analysis on algorithm's procedures, the essence of the proposed DAGH is revealed, which shows how the framework integrates deep hashing and efficient binary code learning seamlessly.
- We test the retrieval performance on three benchmark datasets. Experimental results show that the proposed method performs better than the state-of-the-art hashing methods.

2. Related Work

In the designed model, we use $\mathbf{X}_{all} \in \mathbb{R}^{m\times n_1}$ to denote all the training samples, where $\mathbf{x}_{all,i} \in \mathbb{R}^m$ is the i-th sample vector and n_1 is the total number of samples. The existing deep hashing methods, such as DPSH, DTSH and DSDH, first select a subset from the database to form the training subset and then update the neural network batch by batch. For simplicity, we denote the samples in the subset by $\mathbf{X}_{part} \in \mathbb{R}^{m\times n_2}$, where n_2 is the total number of the samples in the subset. The goal of hashing methods is to learn a set of binary codes for features representation. We use $\mathbf{B}_{all} \in \mathbb{R}^{l\times n_1}$ and $\mathbf{B}_{part} \in \mathbb{R}^{l\times n_2}$ to denote the binary codes of $\mathbf{X}_{all}$ and $\mathbf{X}_{part}$, respectively, where l is the length of bits.

The main purpose of the traditional manifold-based hashing methods is to preserve the latent manifold structure embedding in the high-dimensional data into the low-dimensional binary space [30, 20]. To this end, these methods generally try to solve the following objective function:

$$\begin{aligned} &\min_{\mathbf{B}_{all}} \sum_{i,j=1}^{n_1} ||\mathbf{b}_{all,i} - \mathbf{b}_{all,j}||^2 \mathbf{S}_{ij} \\ &\text{s.t. } \mathbf{b}_{all,i} \in \{-1, 1\}^l \end{aligned} \tag{1}$$

where $\mathbf{b}_{all,i}$ is the i-th column of $\mathbf{B}_{all}$ and $\mathbf{S} \in \mathbb{R}^{n_1\times n_1}$ is the affinity matrix. Problem (1) aims to minimize the distance between $\mathbf{b}_{all,i}$ and $\mathbf{b}_{all,j}$ if $\mathbf{x}_{all,j}$ is the nearest data point of $\mathbf{x}_{all,i}$ in original high-dimensional space. Since constructing the affinity matrix is time-consuming for large-scale dataset, AGH proposes the anchor graph to characterize the local structure [20]. The core idea of anchor graph is to use a small number of anchors to connect the whole dataset so that the similarities between different data points can be computed in an implicit way.

Similar to the problem (1), some recently proposed deep hashing methods minimize the information loss between the deep features and binary codes as follows:

$$\begin{aligned} &\min_{\mathbf{B}_{part},\mathbf{U}} \sum_{i=1}^{n_2} ||\mathbf{b}_{part,i} - \mathbf{u}_i||^2 \\ &\text{s.t. } \mathbf{b}_{part,i} \in \{-1, 1\}^l \end{aligned} \tag{2}$$

where $\mathbf{b}_{part,i}$ is the i-th column of $\mathbf{B}_{part}$ and $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, .., \mathbf{u}_i, .., \mathbf{u}_{n_2}] \in \mathbb{R}^{l\times n_2}$ is the output of the network. Due to the high computation cost, deep hashing methods usually select a subset from the training set for training and it further requires to form mini-batch data to update the network in the inner loop. Therefore, solving problem (2) cannot learn the optimal $\mathbf{B}_{all}$ for retrieval. In next section, we introduce the proposed framework for discriminative binary code learning to overcome this drawback.

3. Deep Anchor Graph Hashing

As shown in problem (2), the previous deep hashing methods directly regress the deep features to the corresponding binary codes to relax the discrete optimization problem. Therefore, in each iteration, only a mini-batch binary code can be learned. As such, these methods need to adopt two steps operation to obtain the optimal binary codes of the entire training set, which will lead to information loss.

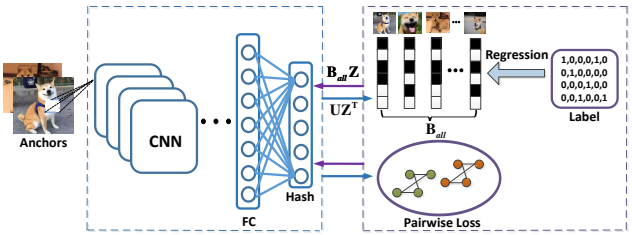


Figure 1. Overview of DAGH. The blue box is the part of neural network and the purple box is the part of objective function. By designing the anchor graph based objective function, the framework can achieve the following two goals. On the one hand, the output of the network and label matrix can be used to generate the binary codes of training set and on the other, the network can obtain more discriminative and effective feedback from the learned bits and pairwise loss.

Besides, the learned hashing function is less discriminative since the network only fits one batch of binary codes in each iteration. In this paper, we hope to integrate the advantages of deep hashing and efficient binary code learning to increase retrieval performance and the generalization ability of the model. Therefore, the problems caused by subset selection and mini-batch operation can be addressed.

3.1. Problem Formulation

We design a joint framework to effectively integrate deep hashing and efficient binary code learning. First, a novel regression term is proposed to establish the connections between the deep features and all the binary codes.

Regression term: We can find the main difference between (1) and (2) is that the distances are measured with or without the affinity matrix. However, the previous deep models cannot directly use all the training data to obtain the binary codes of training set as in (1) because of the mini-batch operation and the high computation cost of deep learning. Therefore, we regard the selected samples as anchors and design the following regression term to minimize the distances between deep features and binary codes:

$$\begin{aligned} \min_{\mathbf{B}_{all},\mathbf{U}} \mathcal{R}_1 &= \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} ||\mathbf{b}_{all,i} - \mathbf{u}_j||^2 \mathbf{Z}_{ij} \\ \text{s.t. } \mathbf{b}_{all,i} &\in \{-1,1\}^l,\ \mathbf{b}_{all}^i \mathbf{1}_{n_1} = 0 \end{aligned} \tag{3}$$

where $\mathbf{b}_{all}^i$ is the i-th row of matrix $\mathbf{B}_{all}$ and $\mathbf{1}_{n_1}$ is n_1-dimensional column vector with elements 1, $\mathbf{Z} \in \mathbb{R}^{n_1 \times n_2}$ is the designed anchor graph. The constraint $\mathbf{b}_{all}^i \mathbf{1}_{n_1} = 0$ is added to make sure that each row has 50% to be -1 and 50% to be 1 so that the bits of binary codes are balanced. The definition of anchor graph $\mathbf{Z}$ in our model is as follows:

$$\mathbf{Z}_{ij} = \begin{cases} \frac{1}{\delta_j}, & \mathbf{b}_{all,i} \text{ and } \mathbf{u}_j \text{ belong to the same class} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where $\mathbf{Z}_{ij}$ is the i-th row and j-th column element of $\mathbf{Z}$ and δ_j is the total number of data points related to $\mathbf{u}_j$. We regard $\mathbf{Z}$ as anchor graph since $n_2 << n_1$ and it is similar to the definition of anchor graph in AGH. By solving (3), the discrete solutions of the training set can be easily obtained. Furthermore, in each iteration, the mini-batch data can approximate to all the related binary codes so that the learned network will become more discriminative. Problem (3) can be regarded as the combination of problems (1) and (2). However, different from these methods, the designed regression term is able to perform efficient binary code learning and deep learning simultaneously. Then, we need to improve the quality of the deep features and binary codes by imposing deep hashing loss and global data information on the model, respectively.

Deep hashing: We hope that the outputs of the network are close to the final binary codes. Therefore, if two deep feature vectors belong to the same class, the inner product of them should be maximized so that the distance between them in binary space can be minimized. Otherwise, the inner product of them should be minimized. The goal can be achieved by solving the following problem:

$$\min_{\mathbf{U}} \mathcal{R}_2 = \sum_{\mathbf{A}_{ij} \in \mathbf{A}} (log(1 + e^{\mathbf{\Theta}_{ij}}) - \mathbf{A}_{ij}\mathbf{\Theta}_{ij}) \tag{5}$$

where $\mathbf{\Theta}_{ij} = \frac{1}{2}\mathbf{u}_i^T \mathbf{u}_j$ and the pairwise relation is given as follows:

$$f(\mathbf{A}_{ij}|\mathbf{U}) = \begin{cases} \sigma(\mathbf{\Theta}_{ij}), & \mathbf{A}_{ij} = 1 \\ 1 - \sigma(\mathbf{\Theta}_{ij}), & \text{otherwise} \end{cases} \tag{6}$$

where $\sigma(\mathbf{\Theta}_{ij}) = \frac{1}{1+e^{-\mathbf{\Theta}_{ij}}}$, the similarity matrix $\mathbf{A} \in \mathbb{R}^{n_2 \times n_2}$ is defined as follows: if $\mathbf{u}_i$ and $\mathbf{u}_j$ belongs to the same class, $\mathbf{A}_{ij} = 1$, otherwise, $\mathbf{A}_{ij} = 0$ and $\mathbf{A}_{ij}$ is the i-th row and j-th column element of $\mathbf{A}$.

Efficient binary code learning: By designing regression term (3), we can fully take advantage of the available discriminative information to improve the quality of the binary codes as well as increase the performance of the network. The common approach is to further classify them with the label matrix as in [25, 16]. However, we hope to solve the model in a more efficient way. Therefore, we introduce the technique in [8] for binary code learning. The discriminant term is as follows:

$$\begin{aligned} \min_{\mathbf{B}_{all},\mathbf{W}} \mathcal{R}_3 &= \sum_{i=1}^{n_1} ||\mathbf{b}_{all,i} - \mathbf{W}^T \mathbf{y}_i||^2 \\ \text{s.t. } \mathbf{b}_{all,i} &\in \{-1,1\}^l,\ \mathbf{b}_{all}^i \mathbf{1}_{n_1} = 0 \end{aligned} \tag{7}$$

where $\mathbf{W} \in \mathbb{R}^{c \times l}$ is regression matrix, $\mathbf{y}_i$ is i-th column of $\mathbf{Y}$ and $\mathbf{Y} \in \mathbb{R}^{c \times n_1}$ is label matrix, i.e., if $\mathbf{b}_{all,j}$ belongs to the i-th class, $\mathbf{Y}_{ij} = 1$, otherwise $\mathbf{Y}_{ij} = 0$, c is the number of classes. Equation (7) aims to learn a discriminative

regression matrix to construct the binary codes bit by bit using the label matrix. Therefore, each class of samples tends to obtain the same binary code. Different from [8], we add the constraint of $\mathbf{b}_{all}^{i}\mathbf{1}_{n_1} = 0$ to generate balanced binary code.

DAGH framework: Finally, by integrating $\mathcal{R}_1$, $\mathcal{R}_2$ and $\mathcal{R}_3$ in one framework, the objective function of DAGH is formulated as follows:

$$\begin{aligned} &\min_{\mathbf{W},\mathbf{B}_{all},\mathbf{U}} \mathcal{L} = \gamma_1\mathcal{R}_1 + \gamma_2\mathcal{R}_2 + \gamma_3\mathcal{R}_3 \\ &\text{s.t. } \mathbf{b}_{all,i} \in \{-1,1\}^l,\ \mathbf{b}_{all}^{i}\mathbf{1}_{n_1} = 0 \end{aligned} \tag{8}$$

where γ_1, γ_2 and γ_3 are weight coefficients. The DAGH framework jointly performs deep hashing and binary code learning so that the exact binary codes of training set can be derived and the network can become more discriminative. The overview of DAGH framework is shown in Figure 1.

3.2. Optimization Algorithm

Since the proposed model contains three variables, we design an iterative algorithm to alternately optimize them. Given $\mathbf{B}_{all}$ and $\mathbf{U}$, we have following subproblem:

$$\min_{\mathbf{W}} \sum_{i=1}^{n_1} ||\mathbf{b}_{all,i} - \mathbf{W}^T\mathbf{y}_i||^2 \tag{9}$$

Problem (9) is a classical linear regression problem. From (9), we can derive:

$$\min_{\mathbf{W}} tr(-2\mathbf{B}_{all}^T\mathbf{W}^T\mathbf{Y} + \mathbf{Y}^T\mathbf{W}\mathbf{W}^T\mathbf{Y}) \tag{10}$$

Taking the partial derivative with respect to $\mathbf{W}$ to be zero, we obtain:

$$\mathbf{W} = (\mathbf{Y}\mathbf{Y}^T)^{-1}\mathbf{Y}\mathbf{B}_{all}^T \tag{11}$$

Note that $(\mathbf{Y}\mathbf{Y}^T)^{-1}\mathbf{Y}$ can be computed in advance. Besides, since the term of $\mathbf{Y}\mathbf{Y}^T \in \mathbb{R}^{c\times c}$ is diagonal matrix for single label dataset and the size of c is usually small, the computational cost can be greatly reduced.

Given $\mathbf{W}$ and $\mathbf{U}$, we have following subproblem:

$$\begin{aligned} &\min_{\mathbf{B}_{all}} \gamma_1\mathcal{R}_1 + \gamma_3\mathcal{R}_3 \\ &\text{s.t. } \mathbf{b}_{all,i} \in \{-1,1\}^l,\ \mathbf{b}_{all}^{i}1_{n_1} = 0 \end{aligned} \tag{12}$$

Problem (12) is a series of regression problems with the discrete constraints. By expanding the formulation and discarding the constant, we derive the following maximization problem:

$$\begin{aligned} &\max_{\mathbf{B}_{all}} tr(\frac{\gamma_1}{\gamma_3}\mathbf{B}_{all}^T\mathbf{U}\mathbf{Z}^T + \mathbf{B}_{all}^T\mathbf{W}^T\mathbf{Y}) \\ &\text{s.t. } \mathbf{b}_{all,i} \in \{-1,1\}^l,\ \mathbf{b}_{all}^{i}\mathbf{1}_{n_1} = 0 \end{aligned} \tag{13}$$

To fulfill the discrete constraints, the optimal $\mathbf{B}_{all}$ can be updated by the following steps. Suppose $\mathbf{Q} = \frac{\gamma_1}{\gamma_3}\mathbf{U}\mathbf{Z}^T + \mathbf{W}^T\mathbf{Y}$ and $\mathbf{q}^i$ is the i-th row vector of $\mathbf{Q}$, we decide the values of binary codes by the following rules:

$$\mathbf{B}_{all,ij} = \begin{cases} 1, & \text{if } j \in \{i\}_{index} \\ -1, & \text{otherwise} \end{cases} \tag{14}$$

where $\mathbf{B}_{all,ij}$ is the i-th row and j-th column element of $\mathbf{B}_{all}$ and $\{i\}_{index}$ is the index set of the first $\frac{n_1}{2}$ maximal elements of $\mathbf{q}^i$.

Given $\mathbf{W}$ and $\mathbf{B}_{all}$, we have following subproblem:

$$\min_{\mathbf{U}} \mathcal{L}(\mathbf{U}) = \gamma_1\mathcal{R}_1 + \gamma_2\mathcal{R}_2 \tag{15}$$

We use back-propagation to update the network [22]. Taking the partial derivative with respect to $\mathbf{U}$, we derive

$$\frac{\partial\mathcal{L}(\mathbf{U})}{\partial\mathbf{U}} = \frac{2\gamma_1}{\gamma_2}(\mathbf{U} - \mathbf{B}_{all}\mathbf{Z}) + \frac{1}{2}\sum_{\mathbf{A}_{ij}\in\mathbf{A}}(\sigma(\mathbf{\Theta}_{ij}) - \mathbf{A}_{ij})\mathbf{U} \tag{16}$$

The details of the algorithm are shown in Algorithm 1.

Algorithm 1 Deep Anchor Graph Hashing (DAGH)

Input: Training data $\mathbf{X}_{all}$, label matrix $\mathbf{Y}$, number of anchors n_2, length of codes l, weight coefficients γ_1, γ_2, γ_3, iteration times T_1 and T_2.

Train:

1: Initialize $\mathbf{B}_{all}$ as arbitrary binary matrix.
2: Start iteration:
For $i = 1 : T_1$ do
  Step 1: $\mathbf{W} = (\mathbf{Y}\mathbf{Y}^T)^{-1}\mathbf{Y}\mathbf{B}_{all}^T$.
  Step 2: Randomly select n_2 data points to form the training subset $\mathbf{X}_{part}$ and compute the corresponding anchor graph $\mathbf{Z}$.
  Step 3: For $j = 1 : T_2$ do
    Select mini-batch data from $\mathbf{X}_{part}$.
    Compute $\mathbf{u}_i$ by forward propagation.
    Compute weight $\mathbf{\Theta}_{ij}$.
    Update neural network by (16).
  end
  Step 4: Obtain $\mathbf{B}_{all}$ according to (14).
end

Output: Hashing function $\psi(\cdot)$ and binary codes $\mathbf{B}_{all}$.

3.3. The Algorithm Mechanism of DAGH

From the algorithm's steps, we can reveal how the framework integrates deep hashing and efficient binary code learning seamlessly. Equation (16) shows that the deep neural network can obtain the feedback from the linear combinations of the binary codes, i.e., $\mathbf{B}_{all}\mathbf{Z}$, where $\mathbf{B}_{all}$ is learned by using all the labeled information. Therefore, the network becomes more discriminative in each mini-batch iteration since all the training samples are involved. On the

other hand, the deep features and label matrix can be jointly used to improve the quality of the binary codes according to the solutions of problem (13). Based on these facts, we can find that the essence of the proposed framework is to fully utilize the global data information to enable deep features and binary codes to influence each other through the regression term (3). As such, the optimal binary codes are derived for retrieval.

3.4. Binary Code Learning

The learned deep neural network can be used to obtain the binary codes of testing samples. That is:

$$\mathbf{b}_{test,i} = sign(\psi(\mathbf{x}_{test,i})) \tag{17}$$

where $\mathbf{x}_{test,i}$ is the i-th testing sample, $\mathbf{b}_{test,i}$ is the corresponding binary code, $\psi(\cdot)$ represents the output of the neural network, $sign(\cdot)$ is the binary function.

4. Experiments

We evaluate the performance of our method on three large-scale datasets, i.e., the CIFAR-10[1], Fashion-MNIST[2] and NUS-WIDE[3] datasets. Some unsupervised and supervised hashing methods are selected for comparison. The unsupervised methods include data-independent method LSH [6], manifold learning method SH [30], AGH [20] and iterative quantization method ITQ [7]. For ITQ, we use Principal Component Analysis (PCA) [13] to reduce the dimensions of data. The supervised methods SDH [25] and its variant FSDH [8] are also added to the experiments. The above methods are based on hand-crafted features. We select some representative deep hashing methods, i.e., DPSH [17], DTSH [29], DSDH [16] and two recently proposed methods Asymmetric Deep Supervised Hashing (ADSH) [12] and method in [5], to compare with our method. Furthermore, to investigate the benefit of our model for directly obtaining the binary codes of training set, we design two models with different binary code learning mechanisms, namely DAGH-1 and DAGH-2. DAGH-1 adopts the neural network to obtain the binary codes. That is:

$$\mathbf{b}_{all,i} = sign(\psi(\mathbf{x}_{all,i})) \tag{18}$$

DAGH-2 represents the proposed method, which directly uses the binary codes derived by Algorithm 1 for retrieval.

4.1. Datasets

Three popular datasets are used to test the performance of different methods.

CIFAR-10 contains 60,000 images belonging to 10 classes. 5,900 images are randomly sampled from each class to establish the training set and the rest images are used as testing data. For hand-crafted features based methods, we use all the 59,000 training samples to learn the hashing function. For deep hashing methods, 5,000 images are selected to form the training subset to save the computational cost. The proposed DAGH also randomly selects 5,000 images to be the anchors in each iteration.

Fashion-MNIST has 10 classes and each class contains 7,000 images in total. The whole dataset is divided into one training set with 60,000 images and one testing set with 10,000 images. Specifically, we randomly select 6,000 and 1,000 images from each class to be the training and testing set, respectively. Similar to the settings on CIFAR-10, the traditional methods SH, AGH, ITQ, SDH and FSDH use the entire training set to train the models and the DPSH, DTSH, DSDH and DAGH select 5,000 images to generate the training subset and anchor points, respectively.

NUS-WIDE is a multi-label dataset. Follow the experimental settings in DPSH, we choose 21 classes from the dataset and each class contains at least 5,000 related images. The final dataset includes over 190,000 images for training and 2,100 images for testing. The deep hashing methods only use 10,500 images as training subset. Since each image contains multiple label, we determine two images belong to the same class when having at least one same label.

For hand-crafted features based methods, we first use the CNN pre-trained on ImageNet [3] to extract the deep features, and then adopt PCA to perform dimensionality reduction. The 1000-dimensional vectors are finally obtained for binary code learning.

4.2. Experimental Settings

To test the performances of different methods with different codes lengths, the number of bits is ranged from $[12, 24, 32, 48]$ on three datasets. For deep hashing methods DPSH, DTSH and DSDH, we re-run the corresponding algorithms by using the codes released by the authors. For fair comparison, the same CNN is applied to all the methods for feature learning. The selected CNN contains seven layers as the structure in [2] and has been pre-trained on the ImageNet. The parameters of different methods are carefully set according to the descriptions of corresponding papers.

We evaluate the retrieval performance of our models and compared baselines with four indicators: Precision rate, Recall rate, F-measure rate and Mean Average Precision (MAP). Specifically, precision and recall are calculated based on 2 hamming distances.

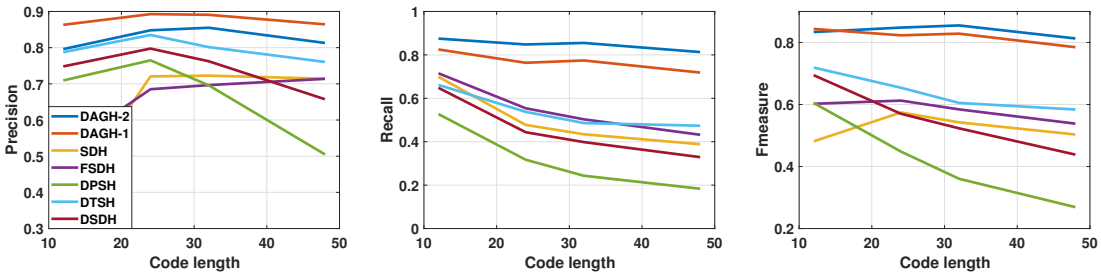
4.3. Discussions

The results on MAP of different methods are shown in Table 1. Since we cannot re-run DTSH on multi-label dataset, the results of DTSH on NUS-WIDE are not present-

[1] http://www.cs.toronto.edu/kriz/cifar.html
[2] https://github.com/zalandoresearch/fashion-mnist
[3] http://lms.comp.nus.edu.sg/research/NUS-WIDE.htm

Table 1. MAP results of different methods on CIFAR-10, Fashion-MNIST and NUS-WIDE datasets with different bits.

Method	CIFAR-10				Fashion-MNIST				NUS-WIDE			
	12	24	32	48	12	24	32	48	12	24	32	48
CNN+LSH	0.152	0.157	0.144	0.163	0.259	0.246	0.274	0.331	0.390	0.422	0.408	0.460
CNN+SH	0.203	0.183	0.179	0.175	0.356	0.316	0.318	0.295	0.419	0.410	0.406	0.432
CNN+AGH	0.277	0.265	0.254	0.232	0.478	0.389	0.347	0.327	0.526	0.511	0.482	0.466
CNN+ITQ	0.216	0.193	0.202	0.197	0.372	0.385	0.365	0.368	0.511	0.522	0.522	0.539
CNN+SDH	0.549	0.669	0.674	0.683	0.629	0.791	0.801	0.807	0.652	0.662	0.652	0.680
CNN+FSDH	0.610	0.657	0.664	0.684	0.777	0.799	0.807	0.811	0.577	0.579	0.580	0.585
DPSH	0.661	0.716	0.723	0.739	0.716	0.800	0.807	0.819	0.682	0.713	0.719	0.726
DTSH	0.770	0.825	0.819	0.834	0.835	0.860	0.861	0.867	–	–	–	–
DSDH	0.735	0.779	0.803	0.816	0.764	0.814	0.799	0.828	0.685	0.710	0.719	0.727
DAGH-1	0.881	0.888	0.900	0.891	0.870	0.882	0.884	0.891	0.707	0.724	0.727	0.733
DAGH-2	**0.934**	**0.933**	**0.934**	**0.932**	**0.932**	**0.938**	**0.937**	**0.937**	**0.760**	**0.789**	**0.793**	**0.802**



(a) (b) (c)

Figure 2. (a) Precisions, (b) Recalls and (c) F-measures obtained by supervised methods on CIFAR-10 dataset.

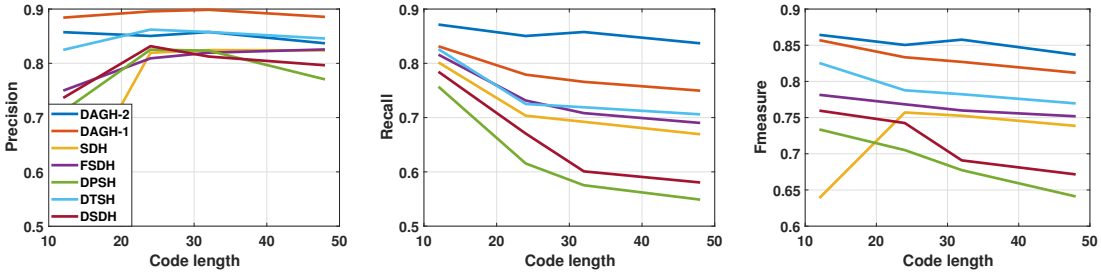
Table 2. MAP results of three deep hashing methods on CIFAR-10 dataset.

Method	12 bits	24 bits	32 bits	48 bits
ADSH	0.889	0.928	0.931	**0.939**
Method in [5]	0.786	0.813	0.821	0.828
DAGH	**0.934**	**0.933**	**0.934**	0.932

ed. It is obvious that the supervised methods can outperform the unsupervised methods. By using the pre-trained CNN to generate the deep features, the SDH and FSDH can also obtain promising performance on Fashion-MNIST dataset with 32 bits. However, the deep hashing methods are better compared to the traditional methods in most cases. In other words, by combing feature learning and binary code learning in an end-to-end framework, the deep hashing methods can greatly improve the retrieval performance. The DAGH-1 is at least 5% and 2% higher than the other deep hashing methods on CIFAR-10 and Fashion-MNIST datasets, respectively. The DAGH-2 is at least 9% and 7% higher than the other deep hashing methods on CIFAR-10 and Fashion-MNIST datasets, respectively. This is because the proposed method fully utilizes the supervised information of the training data to improve the quality of the binary codes. Therefore, the network can obtain more discriminative feedback for training. Since the DAGH-2 obtains the binary codes of training set without information loss, the performance of DAGH-2 is much better than DAGH-1. On NUS-WIDE dataset, the DAGH-1 is slightly better than the DPSH and DSDH. The DAGH-2 is about 6% better than the other deep hashing methods.

In Table 2, two recently proposed methods ADSH and method in [5] are included for comparison. The MAP results of these methods are from [12] and [5], respectively. ADSH can also obtain the binary codes of the entire training set by calculating the inner product between binary codes and deep features. However, ADSH needs relatively high computation cost to optimize the binary codes by using discrete cyclic coordinate descent (DCC) algorithm [25]. The proposed model avoids this problem by designing a regression-based model. Besides, Table 2 shows that the proposed method performs better in most cases.

The results on precision, recall and F-measure of supervised methods on CIFAR-10 and Fashion-MNIST datasets are displayed in Figure 2 and Figure 3, respectively. From the results of DPSH and DSDH, we can find it is challenged to recall the data points of the same class and preserve high



(a) (b) (c)

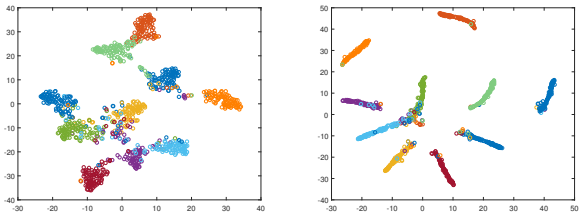
Figure 3. (a) Precisions, (b) Recalls and (c) F-measures obtained by supervised methods on Fashion-MNIST dataset.

Table 3. Training times (in minute) and MAP results of different methods on CIFAR-10 and Fashion-MNIST datasets with 12 bits.

	Method	DPSH	DTSH	DSDH	DPSH-All	DTSH-All	DSDH-All	DAGH
CIFAR-10	Time	58.5m	102.2m	58.9m	842.3m	1210.1m	846.1m	48.5m
	MAP	0.661	0.770	0.735	0.925	0.893	0.931	0.934
F-MNIST	Time	59.2m	89.9m	59.7m	876.9m	1211.8m	859.5m	48.9m
	MAP	0.716	0.835	0.764	0.894	0.909	0.905	0.932

Table 4. Training times for different variables of the proposed model on three datasets with 48 bits.

	$\mathbf{W}$	$\mathbf{B}_{all}$	$\mathbf{U}$	**Total**
CIFAR-10	0.2s	5.8s	49.1m	49.2m
F-MNIST	0.2s	5.9s	49.6m	49.7m
NUS-WIDE	1.1s	1.4m	121.6m	123.0m



(a) (b)

Figure 4. t-SNE representation of real values obtained by (a) DTSH and (b) DAGH.

precision in such small range. The traditional methods SDH and FSDH achieve better performance compared to the DPSH and DSDH. This is because they are more discriminative by utilizing all the training data. However, it requires high computation cost to train DPSH and DSDH with all the samples. The discussions on computation costs of different methods will be given in the following subsection. By combining efficient binary code learning and deep feature learning in one framework, the proposed DAGH can better compress the distances of the similar data points and obtain good retrieval results. As shown in the figures, DAGH-1 outperforms the other methods in terms of precision and DAGH-2 obtains best performance in terms of recall. Besides, Figure 2(c) and 3(c) indicate that DAGH-2 is more effective considering the results of F-measure.

4.4. Computation Cost

Table 3 displays the computation costs of different methods on CIFAR-10 and Fashion-MNIST datasets with 12 bits. It is shown that the proposed method outperforms DPSH, DTSH and DSDH with lower computation cost. Although DTSH performs better than DPSH and DSDH, it needs nearly 1.7 times of training cost. Since the proposed method uses the entire training set to optimize the model, we further test the performance of different deep hashing methods utilizing all the training samples. The corresponding results are listed in the columns of DPSH-All, DTSH-All and DSDH-All, respectively. DPSH-All, DSDH-All and the proposed DAGH obtain similar retrieval performance on CIFAR-10 as shown in the third row. However, DPSH-All and DSDH-All require 17 times more training time than DAGH. The proposed method avoids this problem by taking advantages of anchor graph and efficient binary code learning. Similar conclusions can also be drawn on the Fashion-MNIST dataset. To comprehensively analyze the training times for different variables of the proposed model, we give the detailed training times in Table 4. Compared to the DPSH, the additional computation costs of our method are the optimizations of $\mathbf{W}$ and $\mathbf{B}_{all}$. Since $(\mathbf{Y}\mathbf{Y}^T)^{-1}\mathbf{Y}$ can be pre-computed, the total computation cost of $\mathbf{W}$ and $\mathbf{B}_{all}$ is $O(2lcn_1 + ln_2n_1)$ and linear to the number of samples. Therefore, even if we use the entire training set to optimize the model, the binary codes learning for all samples is

Table 5. MAP results of DAGH with different regularization terms on NUS-WIDE dataset.

	$\mathcal{R}_1$	$\mathcal{R}_1\mathcal{R}_2$	$\mathcal{R}_1\mathcal{R}_3$	$\mathcal{R}_1\mathcal{R}_2\mathcal{R}_3$
12 bits	0.739	**0.772**	0.740	0.760
24 bits	0.748	0.776	0.775	**0.789**
32 bits	0.746	0.780	0.785	**0.793**
48 bits	0.762	0.787	0.790	**0.802**

Table 6. MAP results of different deep hashing methods on CIFAR-10 dataset.

Method	12 bits	24 bits	32 bits	48 bits
DPSH	0.690	0.707	0.723	0.729
DTSH	0.699	0.686	0.711	0.732
DSDH	0.686	0.713	0.701	0.729
DAGH	**0.737**	**0.763**	**0.743**	**0.751**

very efficient. Results in Table 4 also verify this conclusion.

4.5. Visualization of Codes

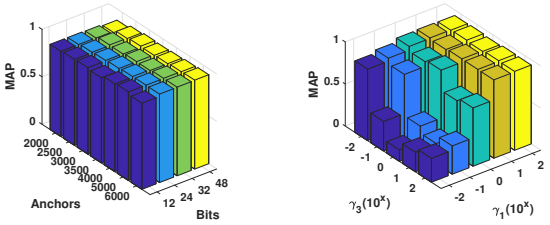
In Figure 4, we display the t-SNE representation [21] of the real values obtained by DTSH and DAGH on CIFAR-10 dataset. Each color denotes a class of samples. To test the generalization abilities of the models, we first learn the 12-dimensional real value vectors of the testing samples by using the learned deep neural network, and then perform t-SNE algorithm to visualize the real values on 2-dimensional spaces. We select the deep hashing method DTSH for comparison based on the results in Table 1 and Figure 2. Intuitively, the proposed DAGH better classifies the binary codes belonging to different classes and compresses the intra-class distances.

4.6. Effectiveness of Regularization terms

Table 5 displays the results on MAP of DAGH with different regularization terms on NUS-WIDE dataset. $\mathcal{R}_1$ represents the original model (3) without any regularization terms, $\mathcal{R}_1\mathcal{R}_2$ adds the deep hashing term to learn compact deep features, $\mathcal{R}_1\mathcal{R}_3$ adds the term of discriminative binary code learning, and the last column presents the results of the proposed framework. We can find that the terms of $\mathcal{R}_2$ and $\mathcal{R}_3$ do not co-operate well with 12 bits. However, as the length of codes increases, both of them can significantly improve the performance of the model, and the proposed framework obtains the best performance as shown in the last column. The experimental results verify the effectiveness of the regularization terms and the framework.

4.7. Retrieval of Unseen Classes

To further verify the performance of supervised hashing methods according to reference [23], we select 7 classes from the CIFAR-10 dataset for the use of training. The remaining 3 classes are used to evaluate the retrieval performance of unseen classes. The MAP results of four deep supervised hashing methods are in Table 6. As is presented in the table, DAGH achieves better results in this case.



(a) (b)

Figure 5. MAPs versus (a) the number of anchors and bits, (b) the variations of γ_1 and γ_3.

4.8. Parameter Sensitivity

Figure 5 shows the performance of DAGH on CIFAR-10 dataset with different parameters. Figure 5(a) shows that the number of anchors has a slight impact on MAP. The results on MAP versus the variations of γ_1 and γ_3 are shown in Figure 5(b). For simplicity, we set $\gamma_2 = 1$ and $l = 24$ in this experiment. It is clear that DAGH does not perform well when γ_1 is small.

5. Conclusion

In this paper, the proposed DAGH framework uses the anchor graph to characterize the similarities between the deep features and all the binary codes. Therefore, the deep features and label matrix can be effectively utilized to generate the binary codes, and inversely, the network can obtain more discriminative feedback from the learned high-quality bits. As such, the data information can be fully exploited to improve the network. Extensive experiments on three datasets show the superior retrieval performance of our method, and verify that even if the proposed model utilizes the entire training set for network optimization, the computation time of our algorithm is still less than the existing deep methods.

6. Acknowledgments

This work was supported by the Natural Science Foundation of China (No. 61802267 and 61732011), the Shenzhen Municipal Science and Technology Innovation Council (JCYJ20180305124834854), the Natural Science Foundation of Guangdong Province (2017A030313367).

References

[1] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Philip S. Yu. Hashnet: Deep learning to hash by continuation. In *ICCV*, pages 5609–5618, 2017.

[2] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *BMVC*, 2014.

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009.

[4] Wei Dong, Zhe Wang, William Josephson, Moses Charikar, and Kai Li. Modeling lsh for performance tuning. In *CIKM*, pages 669–678, 2008.

[5] Chaoyou Fu, Liangchen Song, Xiang Wu, Guoli Wang, and Ran He. Neurons merging layer: Towards progressive redundancy reduction for deep supervised hashing. In *IJCAI*, 2019.

[6] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *VLDB*, pages 518–529, 1999.

[7] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2916–2929, Dec 2013.

[8] Jie Gui, Tongliang Liu, Zhenan Sun, Dacheng Tao, and Tieniu Tan. Fast supervised discrete hashing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(2):490–496, Feb 2018.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[10] Go Irie, Zhenguo Li, Xiao-Ming Wu, and Shih-Fu Chang. Locally linear hashing for extracting non-linear manifolds. In *CVPR*, pages 2123–2130, 2014.

[11] Qing-Yuan Jiang and Wu-Jun Li. Deep cross-modal hashing. In *CVPR*, pages 3270–3278, 2017.

[12] Qing-Yuan Jiang and Wu-Jun Li. Asymmetric deep supervised hashing. In *AAAI*, pages 3342–3349, 2018.

[13] I. T. Jolliffe. Principal component analysis. *Journal of Marketing Research*, 87(100):513, 2002.

[14] Brian Kulis and Kristen Grauman. Kernelized locality-sensitive hashing for scalable image search. In *ICCV*, pages 2130–2137, 2009.

[15] Zhihui Lai, Yudong Chen, Jian Wu, Wai Keung Wong, and Fumin Shen. Jointly sparse hashing for image retrieval. *IEEE Transactions on Image Processing*, 2018.

[16] Qi Li, Zhenan Sun, Ran He, and Tieniu Tan. Deep supervised discrete hashing. In *NIPS*, 2017.

[17] Wu-Jun Li, Sheng Wang, and Wang-Cheng Kang. Feature learning based deep supervised hashing with pairwise labels. In *IJCAI*, pages 1711–1717, 2016.

[18] Haomiao Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen. Deep supervised hashing for fast image retrieval. In *CVPR*, pages 2064–2072, 2016.

[19] Li Liu, Fumin Shen, Yuming Shen, Xianglong Liu, and Ling Shao. Deep sketch hashing: Fast free-hand sketch-based image retrieval. In *CVPR*, pages 2862–2871, 2017.

[20] Wei Liu, Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Hashing with graphs. In *ICML*, pages 1–8, 2011.

[21] Laurens Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[22] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 3(23):533–536, 1986.

[23] Alexandre Sablayrolles, Matthijs Douze, Nicolas Usunier, and Herv Jgou. How should we evaluate supervised hashing? In *ICASSP*, pages 1732–1736, 2017.

[24] Fumin Shen, Xin Gao, Li Liu, Yang Yang, and Heng Tao Shen. Deep asymmetric pairwise hashing. In *MM*, pages 1522–1530, 2017.

[25] Fumin Shen, Chunhua Shen, Wei Liu, and Heng Tao Shen. Supervised discrete hashing. In *CVPR*, pages 37–45, 2015.

[26] Yuming Shen, Li Liu, Fumin Shen, and Ling Shao. Zero-shot sketch-image hashing. In *CVPR*, 2018.

[27] Jun Tang, Ke Wang, and Ling Shao. Supervised matrix factorization hashing for cross-modal retrieval. *IEEE Transactions on Image Processing*, 25(7):3157–3166, July 2016.

[28] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Semi-supervised hashing for large-scale search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(12):2393–2406, Dec 2012.

[29] Xiaofang Wang, Yi Shi, and Kris M Kitani. Deep supervised hashing with triplet labels. In *ACCV*, 2016.

[30] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In *NIPS*, pages 1753–1760, 2009.

[31] Rongkai Xia, Yan Pan, Hanjiang Lai, Cong Liu, and Shuicheng Yan. Supervised hashing for image retrieval via image representation learning. In *AAAI*, pages 2156–2162, 2014.

[32] Xing Xu, Fumin Shen, Yang Yang, and Heng Tao Shen. Discriminant cross-modal hashing. In *ICMR*, pages 305–308, 2016.

[33] Xinyu Yan, Lijun Zhang, and Wu-Jun Li. Semi-supervised deep hashing with a bipartite graph. In *IJCAI*, pages 3238–3244, 2017.

[34] Hanwang Zhang, Fumin Shen, Wei Liu, Xiangnan He, Huanbo Luan, and Tat-Seng Chua. Discrete collaborative filtering. In *SIGIR*, pages 325–334, 2016.

[35] Peichao Zhang, Wei Zhang, Wu-Jun Li, and Minyi Guo. Supervised hashing with latent factor models. In *SIGIR*, pages 173–182, 2014.

[36] Zheng Zhang, Li Liu, Fumin Shen, Heng Tao Shen, and Ling Shao. Binary multi-view clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[37] Fang Zhao, Yongzhen Huang, Liang Wang, and Tieniu Tan. Deep semantic ranking based hashing for multilabel image retrieval. In *CVPR*, pages 1556–1564, 2015.

[38] Yi Zhen and Dit-Yan Yeung. Co-regularized hashing for multimodal data. In *NIPS*, pages 1376–1384, 2012.