Research on Discrete Hash Algorithm Based on Deep Semantics

Qinjin Jia
College of Engineering, Boston University, 8 St Marys St, Boston, MA 02215, USA.
Email: qjia@bu.edu

Abstract. With the rapid growth of image and video data on the network, hash technology has been widely studied in the field of image and video search in recent years. Benefiting from the latest progress in deep learning, deep hash method has achieved good results in image retrieval. However, the previous deep hash method has the limitation that the semantic information is not fully utilized. In this paper, we develop a discrete hash algorithm based on deep supervision, assuming that learning binary code should be an ideal choice of classification. The pair tag information and classified information are used to learn hash codes within a framework. The output of the last layer is restricted to binary code directly, which is rarely studied in deep hash algorithm. Due to the discrete properties of hash codes, the alternate minimization method is used to optimize the target function. The proposed algorithm is proved to be better than the other supervised hash methods in two public image retrieval databases CIFAR-10 and NUS-WIDE.

1. Introduction
Because of the rapid growth of image and video data on the Internet, hash technology has attracted a lot of attention in recent years [1]. Because of its low computational cost and storage efficiency, it is one of the most popular technologies for image or video search. Generally speaking, hashing is used to code high dimensional data into binary codes, while preserving the similarity of images or videos. The existing hash methods can be roughly divided into two categories, data independent methods and data dependent methods.

Data independent methods use random projections to construct hash functions. Locally sensitive hash (LSH) [2] is one of the representative methods. It uses random linear projection to map nearby data to similar binary code. Data dependency method refers to using training data to learn hash functions, which can be further divided into supervised and unsupervised methods. The unsupervised method retrieves the neighborhood under some distance measurements, and iterative quantization (ITQ) [3] is one of the representative unsupervised hash methods, in which the projection matrix is projected through iterative projection and threshold optimization based on a given training sample. In order to make use of the semantic labels of data samples, a supervised hash method named [4, 5, 6] is proposed. The core supervised hashing (KSH) [7] is a well-known method of this type, which learns the hash code by minimizing the Hamming distance between the similar pairs and maximizing the Hamming distance between the dissimilar pairs at the same time.

Recently, a hash method based on deep learning has been proposed to simultaneously learn image representation and hash coding, which has shown superior performance superior to the traditional hash method. The convolution neural network hash (CNNH) [8] is one of the early work of integrating the deep neural network into hash coding. The hash code is composed of two stages, which is used to
learn image representation and hash code. One drawback of CNNH is that learning image representation cannot provide feedback for learning a better hash code. In order to overcome the shortcomings of CNNH, [9] proposed a three rankings loss to capture the relative similarity of the image. Image representation learning and hash coding can benefit from each other within a stage framework.

Although the method based on deep learning has made great progress in image retrieval, the previous deep hash method has the limitation that semantic information is not fully utilized. The recent work attempts to divide the whole learning process into two stream [6] under the framework of multitask learning, which is used to learn hash functions, and the classification flow is used for mining semantic information. Although two stream frameworks can improve retrieval performance [10], classification flow is only used to learn image representation without affecting hash function directly. In this paper, we use CNN to simultaneously learn image representation and hash functions, and the last layer of CNN outputs binary codes directly according to pair tag information and classification information.

The contribution of this work is summarized as follows. (1) The last layer of our method is limited by direct output binary code, learning binary code to maintain a similar relationship and keeping the label consistent, as we know, this is the first deep hash method using pair tag information and classification information to learn a hash code under a stream frame. (2) In order to reduce the quantization error, we maintain the discrete property of hash code in the optimization process, and propose an alternate minimization method, and use the discrete cyclic coordinate descent method to optimize the target function. (3) A large number of experiments show that our method is better than the most advanced method in the reference data set of the image retrieval, which proves the effectiveness of the proposed method.

2. Discrete Hash Algorithm

2.1. Network Structure

This paper utilizes the Convolutional Neural Network (CNN) in [11], namely CNN-F as a deep feature learning part, and replaces the last layer of CNN-F with a fully connected layer to project the output of the second layer into space. The feature learning section contains five convolutional layers ("conv1-conv5") and three fully connected layers ("full6-full8"). The detailed settings of the five convolutional layers are shown in Table 1. In Table 1, "filter size" denotes the number of convolution filters and their reception area size, "stride" denotes the convolution step, "pad" denotes the number of pixels to add to each input size, "LRN" denotes whether to apply Local Response Normalization (LRN), and "pool" denotes the down-sampling factor. Table 2 shows the detailed settings of the three fully connected layers, where the “Settings” show the number of nodes in each layer.

This paper uses the Rectified Linear Unit (ReLU) as the activation function for all the first seven layers. For the last layer, identity function is used as activation function. Figure 1 shows the flowchart of this paper.

Table 1. The Settings of Convolutional layers

| Layer | Settings | Filter size | Stride | Pad | LRN | Pool |
|-------|----------|-------------|--------|-----|-----|------|
| Conv1 |          | 64×11×11    | 4×4    | 0   | Yes | 2×2  |
| Conv2 |          | 256×5×5     | 1×1    | 2   | Yes | 2×2  |
| Conv3 |          | 256×3×3     | 1×1    | 1   | No  | -    |
| Conv4 |          | 256×3×3     | 1×1    | 1   | No  | -    |
| Conv5 |          | 256×3×3     | 1×1    | 1   | No  | 2×2  |
Table 2. The Settings of Fully connected layers

| Layer | Settings |
|-------|----------|
| full6 | 4096     |
| full7 | 4096     |
| full8 | Hash code length k |

2.2. Objective function section

For all points, given a binary code \( \beta = \{ b_i \}_{i=1}^{n} \), we can define the likelihood between label pairs \( S = \{ s_y \} \) as the likelihood of LFH. The formula is as follows:

\[
p(s_y | \beta) = \begin{cases} 
\sigma(\Omega_y), & s_y = 1 \\
1 - \sigma(\Omega_y), & s_y = 0 
\end{cases}
\]

(1)

where \( \Omega_y = \frac{1}{2} b_i^T b_j \), \( \sigma(\Omega_y) = \frac{1}{1 + e^{-\Omega_y}} \), please note \( b_j \in \{-1, 1\}^{c} \).

By observing the negative log-likelihood of the paired labels in S, we can get the following optimization problems:

\[
\min_{\beta} J_1 = -\log p(S | \beta) = -\sum_{s_y \in S} \log p(s_y | \beta) = -\sum_{s_y \in S} \left( s_y \Omega_y - \log \left( 1 + e^{\Omega_y} \right) \right)
\]

(2)

It is easy to find that the above optimization problem can make the Hamming distance between two similar points as small as possible while making the Hamming distance between two dissimilar points as large as possible. This exactly matches the goal of a supervised hashing algorithm with paired labels.

The problem in formula (1) is a discrete optimization problem, which is difficult to solve. LFH solves this problem by relaxing \( \{ b_i \} \) directly from discrete to continuous pairs, which may not achieve satisfactory performance.

In this paper, we design a new strategy that can solve this problem in a discrete manner in formula (1). First, we reformulate the problem in formula (1) as the following equivalent problem:

\[
\min_{\beta, u} J_2 = -\sum_{s_y \in S} \left( s_y \Theta y - \log \left( 1 + e^{\Theta y} \right) \right)
\]

(3)

s.t. \( u_i = b_i \), \( \forall i = 1, 2, \cdots, n \)

\( u_i \in \mathbb{R}^{c} \), \( \forall i = 1, 2, \cdots, n \)

\( b_i \in \{-1, 1\}^{c} \), \( \forall i = 1, 2, \cdots, n \)

where \( \Theta y = \frac{1}{2} u_i^T u_j \), \( u = \{ u_i \}_{i=1}^{n} \).

To optimize the problem in formula (2), we can optimize the following regularization problem by moving the equality constraint in formula (2) to the regularization term:

\[
\min_{\beta, u} J_3 = -\sum_{s_y \in S} \left( s_y \Theta y - \log \left( 1 + e^{\Theta y} \right) \right) + \eta \sum_{i=1}^{n} \| b_i - u_i \|_2^2
\]

(4)

where \( \eta \) is the regularization term (hyper-parameter).

2.3. Model solving section

In order to integrate the above feature learning part and objective function part into the end-to-end framework, we set as followings:
where $\theta$ denotes all the parameters of the seven layers in the feature learning section, $\phi(x; \theta)$ denotes the output of the fully connected seven layers related to the point $x$, $W \in R^{4096 \times c}$ denotes the weight matrix, and $v \in R^{c \times 1}$ denotes the bias vector. This means that we integrate the feature learning part and the objective function part into the same frame through a fully connected layer, with the weight matrix $W$ and the bias vector $v$. After integrating these two parts, the learning problem becomes:

$$
\min_{\beta, W, v, \theta} J = -\sum_{x \in S} (s_y \Theta_y - \log(1 + e^{\Theta_y})) + \eta \sum_{i=1}^{n} \|b_i - (W^T \phi(x; \theta) + v)\|_2^2
$$

As a result, we obtain an end-to-end deep hash model, which we call DPSH. The model can simultaneously perform feature learning and hash code learning in the same framework.

2.4. Parameter learning

In the convolutional neural network, learning parameters include $W$, $v$, $\theta$, and $\beta$. We use a mini-batch-based strategy for learning. More specifically, in each iteration, we sample small batches of points from the entire training set and then learn from these sampling points.

We designed an alternate learning method. That is to say, we optimize one parameter with other parameters fixed.

$b_i$ can be directly optimized as follows:

$$
b_i = \text{sgn}(u_i) = \text{sgn}(W^T \phi(x; \theta) + v).
$$

For other parameters $W$, $v$ and $\theta$, we use Back Propagation (BP) for learning. Specifically, we can use the following formula to calculate the derivative of the loss function for $u_i$:

$$
\frac{\partial J}{\partial u_i} = \frac{1}{2} \sum_{j, y \in S} (a_{ij} - s_{ij}) u_j + \frac{1}{2} \sum_{j, y \in S} (a_{ji} - s_{ji}) u_j + 2\eta (u_i - b_i),
$$

where $a_{ij} = \sigma\left(\frac{1}{2} u_i^T u_j\right)$.

Then we can use the back propagation to update the parameters $W$, $v$ and $\theta$:

$$
\frac{\partial J}{\partial W} = \phi(x; \theta) \left(\frac{\partial J}{\partial u_i}\right)^T,
$$

$$
\frac{\partial J}{\partial v} = \frac{\partial J}{\partial u_i},
$$

$$
\frac{\partial J}{\partial \phi(x; \theta)} = W \frac{\partial J}{\partial u_i}.
$$

The above formula briefly summarizes the entire learning algorithm.

3. Experimental data and analysis

This paper uses two widely used image datasets to evaluate the proposed method. They are CIFAR-10 [4] and NUS-WIDE [5]. The CIFAR-10 dataset contains 60,000 images and is manually sorted to 10 categories, including "aircraft", "car", "bird", "cat", "deer", "dog", "frog", "horse" and "truck". This is a single-label dataset. The size of each image is 32 x 32 pixels. If two images share the same label, they are considered similar, that is to say, they belong to the same category. Otherwise, they are considered dissimilar. First, we adjust the size of all images in the two datasets to 224 x 224 pixels, and then use the original image pixels directly as the input to the deep hash method. The deep hash method uses the same pre-trained CNN-F model on ImageNet for feature learning. We implement this
model on MatConvNet and modify the small batch size to 128 using the cross-validation strategy, meanwhile, adjust the learning rate from 10^-6 to 10^-2. In addition, we set the weights decay as 5×10^-4 so as to avoid overfitting.

In the experiments, the supervisory information is defined based on whether two data points share at least one category label. We evaluate our method and other comparison methods by performing hamming ranking tasks and hash lookup tasks. We show the average accuracy (mAP) of the hamming sorting task. Specifically, given a query \( x_q \), its average accuracy (AP) can be calculated by the following formula:

\[
AP(x_q) = \frac{1}{R_k} \sum_{m=1}^{R_k} P(m)I(m)
\]

where \( R_k \) is the number of related samples, \( P(m) \) is the precision of the cutoff value \( m \) in the returned sample list, and \( I(m) \) is an indicative function. If the \( m \)th returned sample is \( x_q \), then \( I(m) \) is 1, otherwise, \( I(m) \) is 0. Given a \( Q \) query, this article can calculate the mAP as follows:

\[
mAP = \frac{1}{Q} \sum_{q=1}^{Q} AP(x_q)
\]

Because NUS-WIDE is relatively large, the mAP value on NUS-WIDE is calculated based on the first 5000 returned nearest neighbors. The mAP values for other datasets are calculated based on the entire search set.

**Table 3. Performance Comparison of Different Hash Algorithms on CIFAR10 Database**

| Method    | 16 bits(%) | 24 bits(%) | 32 bits(%) | 48 bits(%) | 64 bits(%) |
|-----------|------------|------------|------------|------------|------------|
| Our Method| 70.34      | 71.87      | 72.55      | 72.64      | 73.69      |
| DSH       | 68.04      | 68.64      | 69.02      | 69.43      | 69.86      |
| COSDISH   | 66.17      | 66.41      | 66.62      | 66.86      | 66.96      |
| FastH     | 55.67      | 55.86      | 56.20      | 56.44      | 56.64      |
| LFH       | 42.55      | 43.02      | 43.76      | 44.44      | 45.07      |
| ITQ       | 32.95      | 33.66      | 34.09      | 33.98      | 33.89      |
| LSH       | 13.59      | 13.78      | 13.93      | 14.44      | 14.46      |

**Table 4. Performance Comparison of Different Hash Algorithms on NUS-WIDE Database**

| Method    | 16 bits(%) | 24 bits(%) | 32 bits(%) | 48 bits(%) | 64 bits(%) |
|-----------|------------|------------|------------|------------|------------|
| Our Method| 78.45      | 79.32      | 80.89      | 81.45      | 81.55      |
| DSH       | 74.43      | 75.76      | 76.75      | 77.56      | 78.05      |
| COSDISH   | 70.26      | 71.95      | 72.06      | 73.94      | 74.82      |
| FastH     | 35.32      | 36.28      | 38.64      | 39.11      | 42.63      |
| LFH       | 20.49      | 23.65      | 25.62      | 26.96      | 28.04      |
| ITQ       | 18.54      | 20.85      | 23.04      | 25.43      | 26.75      |
| LSH       | 16.65      | 18.54      | 20.76      | 23.12      | 25.34      |

Table 3 and Table 4 respectively list the MAP results of all the methods of CIFAR-10 and NUS-WIDE under the experimental setup. As you can see from Table 1, the proposed method is obviously superior to the traditional hash method on the CIFAR-10 dataset, and our method's MAP results are more than two times more than LFH, FastH, and ITQ. In addition, most deep hashing methods
perform better than traditional hashing methods. Compared with DSH, our method further improves the performance of 3 to 7%. These results prove that learning hash functions and classifiers in a streaming framework can improve retrieval performance. In the NUS-WIDE dataset, the difference between the deep hashing method and the traditional hash method is also very large, which is the same as the CIFAR-10 dataset. It further illustrates that the performance of a new deep semantic discrete hash algorithm is better than that of other comparison methods.

4. Conclusions
In this paper, we propose a new depth semantic discrete hash algorithm, which directly restricts the output of the last layer to binary code, and the pair tag information and classification information are used to learn hash codes in a stream framework. Due to the discrete properties of hash codes, we derive an alternate minimization method to optimize the loss function. A large number of experiments show that our method performs better than the most advanced method in the datum image retrieval data set.

5. References
[1] CHEN Ya-xiong, HUANG Zhang-chang, FENG Lei. Image compression algorithm based on singular value decomposition and Contourlet transform [J]. Application Research of Computers, 2017, 34 (1): 317-320.
[2] Uhlmann J K. Satisfying General Proximity / Similarity Queries with Metric Trees [J]. Information Processing Letters, 1991, 40(4):175-179.
[3] Gong Y, Lazebnik S, Gordo A, et al. Iterative Quantization: A Procrustean Approach to Learning Binary Codes for Large-scale Image Retrieval. [C]. IEEE Conference on Computer Vision and Pattern Recognition, 2013:817-824.
[4] P. Zhang, W. Zhang, W.-J. Li, and M. Guo. Supervised hashing with latent factor models. In SIGIR, pages 173–182, 2014.
[5] G. Lin, C. Shen, Q. Shi, A. van den Hengel, and D. Suter. Fast supervised hashing with decision trees for high-dimensional data. In CVPR, pages 1963–1970, 2014.
[6] K. Lin, H.-F. Yang, J.-H. Hsiao, and C.-S. Chen. Deep learning of binary hash codes for fast image retrieval. In CVPR, pages 27–35, 2015.
[7] Jiang Q Y, Li W J. Scalable Graph Hashing with Feature Transformation [C]. International Conference on Artificial Intelligence, 2015:2248-2254.
[8] Abadi M, Agarwal A, Barham P, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems [J]. arXiv preprint arXiv:1603.04467, 2016.
[9] Chen Fei, Lu Shaohe, Li Jun, et al. Multi-label image retrieval based on target extraction and hashing [J]. Journal of Image and Graphics, 2017, 22 (2): 232-240.
[10] H. Liu, R. Wang, S. Shan, and X. Chen. “Deep supervised hashing for fast image retrieval,” in IEEE Conference on Computer Vision and Pattern Recognition, 2016.
[11] Z. Zhang, Y. Chen, and V. Saligrama. Efficient training of very deep neural networks for supervised hashing. In CVPR, pages 1487–1495, 2016.
[12] W.-C. Kang, W.-J. Li, and Z.-H. Zhou. “Column sampling based discrete supervised hashing,” in AAAI Conference on Artificial Intelligence, 2016, pp. 1230–1236.