Deep Supervised Hashing for Fast Multi-Label Image

Qian Ying^{1,*}, Ye Qingqing²

¹Laboratory of Graphics-Image and Multimedia, Chongqing University of Posts and Telecommunications, China

²Laboratory of Graphics-Image and Multimedia, Chongqing University of Posts and Telecommunications, China

Abstract. In this paper, most of the existing Hashing methods is mapping the hand extracted features to binary code, and designing the loss function with the label of images. However, hand-crafted features and inadequacy considering all the loss of the network will reduce the retrieval accuracy. Supervised hashing method improves the similarity between sample and hash code by training data and labels of image. In this paper, we propose a novel deep hashing method which combines the objective function with pairwise label which is produced by the Hamming distance between the label binary vector of images, quantization error and the loss of hashing code between the balanced value as loss function to train network. The experimental results show that the proposed method is more accurate than most of current restoration methods.

1 INTRODUCTION

Researchers proposed many efficient retrieval technology in the past ten years, the most successful methods including image retrieval method based on tree, retrieval method based on Hashing which's representative method is a locality sensitive hashing (LSH)[1] and image retrieval method based on vector quantization. Compared with other methods, hashing method have the high efficiency in Hamming distance calculation and the advantages of storage space, so hashing method is very popular in large-scale similar image retrieval.

The Semantic Hashing method based on deep learning is proposed by the Hinton research group in 2009 which opened the door of the deep hashing method[2]. Then, inspired by the powerful ability of the CNN (convolutional neural network)[3], most of researchers apply the CNN based deep hashing method in many fields. Hashing method is divided into two categories: data independent methods and data dependent methods. The first method is randomly map data into binary codes. The typical data independent methods are locality sensitive hashing (LSH)[4] and other modified versions of LSH. The second method uses the trained hash function to map sample to binary code. Compared with the first category, the second way can generate shorter code to get higher accuracy.

The transformation of high-dimensional image features into low dimensional binary hash codes leading the gap between image semantics and hashing code. Therefore, a lot of research is still needed. In this paper, a hashing method for learning by using multi-labels as supervised information is proposed on the basis of many previous methods. The method is combined with deep learning and hashing method, forming the deep

supervised hashing method for multi-label images. The basic ideas of the method:

- Set the label of the images into a set of codes. According to the Hamming distance between any pairwise images, we get the pairwise labels, and then get the label matrix, which simplifies the multiple labels of the image, and is more convenient as supervised information;
- Design a loss function in this paper which contains three components: the difference between hash and image semantics, quantization error when image features are quantized into hash codes, the balance rate for each bit as 0 or 1;
- Add the hidden layer with the sigmoid function in the model which makes the input of hashing method is closer to 0 or 1.

2 RELATED WORKS

Hashing method based on learning: The learning-based hashing methods can be classified into three categories: unsupervised, semi-supervised and supervised. For the first category, label information of the training data is not required. Typical unsupervised hashing method has spectral hashing (SH)[5], iterative quantization (ITQ)[6] and so on. Supervised hashing method uses label of images as supervised information to learn more effective hashing function. Supervised information of hashing method is divided into three types: pointwise, pairwise and list-wise. Pointwise and pairwise convert sorting problems into regression, classification, or orderly classification problems. The typical Pointwise method has supervised discrete hashing (SDH)[7]. And the typical pairwise method has Sequential projection learning for hashing (SPLH)[8], Fast supervised hashing (FastH)[9], Latent factor hashing (LFH)[10],

Convolutional Neural Network Hashing (CNNH)[11], Deep supervised hashing (DPSH)[12] and Supervised semantics-preserving deep hashing (SSDH)[13]. The typical list-wise method has Ranking-based supervised hashing [14], Ranking preserving hashing [15] and other methods.

Deep learning: Deep learning is to train data features by building machine learning framework and mass training data to improve the accuracy of classification or prediction. The loss function is the key of deep learning. We should design the loss function as much as possible including all error, and then gain minimum value through the training network. The researchers should design the loss function according to their model. Deep semantic ranking hashing (DSRH)[16] use the order of image correlation from large to small as supervised information. Deep similarity comparison hashing (DSCH)[17] using weight Hamming distance replace standard Hamming distance.

3 Model and learning

Recently, a new hashing method CNNH predicts the code by convolution neural network (CNN) and cross entropy loss. Compared to the previous method, CNNH's performance improved. After that, Deep Neural Network Hashing (DNNH)[18] and DPSH developed an end-to-end model which can update the binary code by the learned image and better display the ability of deep learning. This section introduces the model which is an end-to-end framework combining feature learning and hashing code learning.

Fig 1. Deep learning network.

Let $X = \{x_i\}_{i=1}^N$ be the training set which contains N samples, where x_i is the i th sample in X . Assume there are K labels, let $L = \{l_i\}_{i=1}^K$ be the K -bit binary codes vector of labels of images, where $l_i \in \{0, 1\}$ indicates whether the label of the current image is existing in the total label. We should convert the multi label to the pairwise label. Let $L_i = \{l_{i1}, l_{i2}, \dots, l_{ik}\}$ and $L_j = \{l_{j1}, l_{j2}, \dots, l_{jk}\}$ as the label vector of i th and j th sample in X , and the pairwise label of i th and j th sample is computed as follow:

$$s_{ij} = (\text{sgn}(\sigma(K - d(L_i, L_j) - 0.5) + 1) / 2) \quad (1)$$

where $d(L_i, L_j)$ is the Hamming distance of L_i and L_j , and $\sigma(\cdot) = 1/(1 + \exp(-z))$ is the sigmoid function, with z a real value. The range of sigmoid function from 0

to 1 which indicates the output of sigmoid function can express the probability that two samples are similar. Then sign function is used to transform probability to 0 or 1. If s_{ij} is 1, then i and j are similar. If 0, it is not similar.

Let $S = \{s_{ij}\}$ as the pairwise label matrix.

Figure 1 show the end to end deep learning framework based on this method. The framework is composed of two parts: convolutional neural network for acquiring image features and loss function modules for frame learning.

3.1 Feature learning part

There are many general frameworks, such as AlexNet and VGG, which integrate deep learning and the creation of hashing function, and make it more convenient to use. This paper introduces the feature learning part based on AlexNet.

This model consists of two CNN, which have the same structure and share the weight of each other. Each network has 5 convolution layers (F_{1-5}) and 2 fully-connected layer (F_{6-7}) as same as the AlexNet. In the convolutional layers, units are organized into feature maps and are connected locally to patches in the outputs of the previous layer. The full connection layer maps the "distributed feature representation" to the space of label sample and as the classifier in network. F_{1-2} and F_{5-7} has the maximum pooling which get the maximum value of the previous layer as output to reduce dimension of feature and preserved the important feature of image. Each convolution layer and the fully connected layer contain the ReLu (Rectified Linear Units) activation function, which makes the training faster.

Let θ are the weight of the feature learning part, x_i, x_j as the input of two networks, the output of F_7 respectively is $a_i^7 = \phi(x_i; \theta)$ and $a_j^7 = \phi(x_j; \theta)$, which will input the latent layer to predict the hashing function by pairwise label matrix.

3.2 Loss Function part

The process of learning is design the appropriate loss function, and then optimizes the weight of network by the stochastic gradient descent (SGD). As shown in Figure 1, the loss function is made up of two parts: the target loss function and the loss function about hashing code. The target loss function is to determine whether the two images are similar by the Hamming distance of the hashing code of the two images. The loss function about hashing code is the loss function set to generate effective hashing codes. The loss function about hashing code contains two parts: the difference between the input of hash function and 0.5 and the probability of each bit of hashing code be 0 or 1.

a_i^7 and a_j^7 which is the output of F_7 input the loss function part, the first layer of this part is fully-connected layer with sigmoid function which make the output closer to 0 or 1.

3.2.1 The Target Loss Function

The output of sigmoid function with the input a_i^7 is $a_i^H = \sigma(a_i^7 W^T + v)$, where $W^T \in R^{d \times K}$ is the weight between F_7 and the latent layer, v is the bias. And the output of latent as follows:

$$b_i = h(a_i^7) = (\text{sgn}(\sigma(W^T a_i^7 + v) - 0.5) + 1) / 2 \quad (2)$$

The hashing code of all images is combined as a binary matrix $B = \{b_i\}_{i=1}^N$. Because discrete optimization is a NP problem, so the target function with pairwise labels can define as LFH which use $u_i = W^T a_i^7 + v$ replace b_i :

$$\min_{B,U} J_1 = - \sum_{s_{ij} \in S} (s_{ij} \Theta_{ij} - \log(1 + e^{\Theta_{ij}})) \quad (3)$$

where $\Theta_{ij} = \frac{1}{2} u_i^T u_j$, $U = \{u_j\}_{j=1}^N$. And we can add the equality constraints as regularization term in (4):

$$\min_{B,U} J_1 = - \sum_{s_{ij} \in S} (s_{ij} \Theta_{ij} - \log(1 + e^{\Theta_{ij}})) + \eta \sum_{i=1}^N \|b_i - u_i\|_2^2 \quad (4)$$

3.2.2 The Loss Function about Hashing Code

Beside the target loss function, we need to optimize the effectiveness of hashing code. The optimization of hashing code consists of two parts: the quantization error between the output of sigmoid function and hashing code and the optimization for keeping the hashing code's balance.

The hashing code is binary code which is generated by images feature through the latent layer, but the range of feature is very large, therefore the optimization of quantization error can make the output of sigmoid function more close to 0 or 1. So the first part of the optimization of hashing code can be maximize the sum of squared errors between the latent layer activations and 0.5, that is $\sum_{i=1}^N \|a_i^H - 0.5e\|^2$.

How to ensure the balance of the hashing code, the best way is the half of bits of all the hashing code is 0, and other bits is 1. We use optimize the model by SGD which is randomly divide all training data into multiple batch, and it is difficult for each batch to keep the hashing code balance. Given an image I^n and a_i^H is a discrete probability distribution on $\{0,1\}$. We hope that the probability of 0 and 1 randomly generated is equal, so the second part of the optimization of hashing code can be $\sum_{i=1}^N (\text{mean}(a_i^H) - 0.5)^2$, where $\text{mean}(\cdot)$ represents the average value of all elements in the vector calculation. This method helps to generate the same number of 0 and 1 for learning sample. At the same time, the minimum Hamming distance between two hash codes with the same number of 0 and 1 is changed to 2, which makes the hash code more separated.

Finally, the two constraints are combined to make each bit of a_i^H have the probability of 50% be 1 or 0:

$$\min_w J_2 = \arg \min_w -E_1(W) + E_2(W) \quad (5)$$

where $p \in \{0,1\}$. The optimization of (6) can optimize most of loss point in the network, and we can gain the

effective hashing code that is similar to the image semantics by minimize the loss function.

3.2.3 Learning

The last loss function combined all the loss term which is proposed above:

$$\min_{W,v} J = J_1 - E_1(W) + E_2(W) \quad (6)$$

where $a_i^H = \sigma(u_i)$. We optimize a parameter and other parameters are fixed in the backward learning. The parameters that need to be optimized by BP learning are W , V and θ . First differentiating u_i :

$$\frac{\partial J}{\partial u_i} = \frac{1}{2} \sum_{j: s_{ij} \in S} (a_{ij} - s_{ij}) u_j + \frac{1}{2} \sum_{j: s_{ji} \in S} (a_{ji} - s_{ji}) u_j + 2\eta(u_i - b_i) - \frac{2}{K} \frac{\partial a_i^H}{\partial u_i} \sum_{i=1}^N (a_i^H - 0.5e) + 2 \frac{\partial a_i^H}{\partial u_i} \sum_{i=1}^N |\text{mean}(a_i^H) - 0.5| \quad (7)$$

$$\text{where } \frac{\partial a_i^H}{\partial u_i} = \frac{\exp(-u_i)}{(1 + \exp(-u_i))^2}$$

Then, we can update W , v and θ by u_i :

$$\begin{aligned} \frac{\partial J}{\partial v} &= \frac{\partial J}{\partial u_i} \\ \frac{\partial J}{\partial v} &= \frac{\partial J}{\partial u_i} \\ \frac{\partial J}{\partial \Phi(x_i; \theta)} &= W \frac{\partial J}{\partial u_i} \end{aligned} \quad (8)$$

4 Experiment

4.1 Dataset and equipment

The CIFAR-10 dataset contains 60,000 color images of size 32×32 , which can be divided into 10 categories (6,000 images for each category). The whole dataset is divided into 50,000 training samples and 10,000 test images.

The NUS-WIDE contains 269,684 images and 81 labels, which is a dataset with multiple labels per images.

The method of this paper compared with other methods in the experiment, which are divided into 4 categories:

- Traditional data-dependent methods with hand-crafted features (512-dimensional GIST descriptor), including SH, ITQ, SPLH, FastH, LFH, SDH.
- Deep hashing methods with pairwise labels, including CNNH, DPSH.
- Deep hashing methods with triplet labels, including DSRH, DSCH.

All experiments were implemented in MATLAB using MatConvNet toolbox with Intel(R) HD Graphics 4600.

4.2 Evaluation protocols

we use two evaluation metrics to evaluate performance of all the above methods. These evaluation metrics represent the difference performance of these hashing method:

- Mean average precision (mAP): We rank all the images according to their Hamming distances to the

query and compute the mAP. mAP is used to measure the accuracy of hashing methods.

- Mean training speed (MTS, images/s): How many images can be trained per second for the method.

4.3 Experimental Results

In the first experiment, we randomly select 1000 images (100 images per class) as the query set in CIFAR-10. For the supervised methods, we randomly select 5000 images (500 images per class). And we randomly sample 2100 query images from 21 most frequent labels (100 images per class) in NUS-WIDE. For the supervised methods, we randomly select 500 images per class as training set. For the unsupervised methods, we use the rest images in CIFAR-10 and NUS-WID as the training set.

The result of first experiment is shown in table 1 and table 2. As shown in Table 1, our method outperforms all other baselines, including data-dependent methods with hand-crafted features and deep hashing methods with pairwise or triplet labels. And we can see the training speed of our method better than DPSH which is foundation of our method in Table 2. Through the analysis of Table 1 and Table 2, we can found not only in the mAP but also in the training speed our method has a better perform compared DPSH.

Table 1. mAP of Hamming ranking for different length

method	CIFAR-10				NUS-WIDE			
	12-bits	24-bits	32-bits	48-bits	12-bits	24-bits	32-bits	48-bits
Ours	0.745	0.762	0.785	0.797	0.810	0.845	0.865	0.882
DPSH	0.713	0.727	0.744	0.757	0.794	0.822	0.838	0.851
NINH	0.552	0.566	0.558	0.581	0.674	0.697	0.713	0.715
CNNH	0.439	0.476	0.472	0.489	0.611	0.618	0.625	0.608
FastH	0.305	0.349	0.369	0.384	0.621	0.650	0.665	0.687
SDH	0.285	0.329	0.341	0.356	0.568	0.600	0.608	0.637
LFH	0.176	0.231	0.211	0.253	0.571	0.568	0.568	0.585
SPLH	0.171	0.173	0.178	0.184	0.568	0.589	0.597	0.601
ITQ	0.162	0.169	0.172	0.175	0.452	0.468	0.472	0.477
SH	0.127	0.128	0.126	0.129	0.454	0.406	0.405	0.400

Table 2. MTS of our method and DPSH in CIFAR-10

method	MTS	The highest speed	The lowest speed
Ours	30	35	25
DPSH	25	28	20

Table 3. mAP of hashing method based on pairwise label and ranking label

method	CIFAR-10				NUS-WIDE			
	12-bits	24-bits	32-bits	48-bits	12-bits	24-bits	32-bits	48-bits
Ours	0.811	0.823	0.830	0.844	0.753	0.761	0.770	0.778
DPSH	0.763	0.781	0.795	0.807	0.716	0.722	0.736	0.741
DSCH	0.609	0.613	0.617	0.620	0.592	0.597	0.611	0.609

DSRH	0.608	0.611	0.617	0.618	0.609	0.618	0.621	0.631
------	-------	-------	-------	-------	-------	-------	-------	-------

Consequently, our method can get more effective hashing code by the modified framework and the loss function which can enhance the speed of generating hashing code and strengthen the ability of distinguish similar or dissimilar hashing code.

Table 3 is the results of the second experiments which is the comparison between hashing method based on pairwise labels and ranking labels. Because of the training data and query data is different from the first experiment. We randomly select 10000 images (1000 images per class) as the query set in CIFAR-10, and the rest as training set. Meantime, we randomly select 2100 images (100 images per class) as the query set, the rest of NUS-WIDE as the training set. our method and DPSH based on pairwise label, others based on ranking labels. We can see our method and DPSH perform better than other methods, and our method give significant improvements over the state-of-the-art deep hashing methods across different number of bits. Therefore, our method can effectively improve the speed and precision of image retrieval systems.

5 Conclusions

This paper proposed a new learning hashing method in the end-to-end model which combines the target loss function and the loss function of hashing code. Our method not only carry on feature learning and hashing code learning simultaneously, but also we can get better effective hashing code. Experiments on specific datasets show that our method improves the accuracy of image retrieval and the speed of training models compared with others.

References

1. DATAR M, IMMORLICA N, INDYK P, J. Proc. Symp. Comput. Geom (2004)
2. DONG W, CHARIKAR M, LI K. J. Proc. ACM SIGIR Conf. Res. Develop. Inf. Retr., 123–130 (2008)
3. A. Krizhevsky, I. Sutskever, and G. E. Hinton. J. In Proc. Adv. Neural Info. Process. Syst. (NIPS) (2012)
4. Andoni A, Indyk P. C. *Foundations of Computer Science.FOCS '06*. IEEE Symposium on. IEEE, 459-468 (2006)
5. Weiss Y, Torralba A, Fergus R. C. *International Conference on Neural Information Processing Systems*. Curran Associates Inc. 1753-1760 (2008)
6. Lazebnik S. C. *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 817-824 (2011)
7. Shen F, Shen C, Liu W, et al. C. *IEEE Conference on Computer Vision and Pattern Recognition*. 37-45 (2016)
8. Wang J, Kumar S, Chang S F. C. *International Conference on International Conference on Machine Learning*. Omnipress. 1127-1134 (2010)

9. Lin K, Yang H F, Hsiao J H, et al. C. IEEE. 27-35 (2015)
10. Zhang P, Zhang W, Li W J, et al. M. *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2014)
11. Xia R, Pan Y, Lai H, et al. C. *AAAI Conference on Artificial Intelligence* (2012)
12. Li, Wu-Jun, Wang, Sheng, Kang, Wang-Cheng. J. *International Joint Conference on Artificial Intelligence*. 1711-1717 (2015)
13. Yang H F, Lin K, Chen C S. J. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, PP(99):1-1 (2015)
14. Wang J, Liu W, Sun A X, et al. J. *IEEE International Conference on Computer Vision*. 3032-3039 (2013)
15. Li W J, Wang S, Kang W C. C. *International Joint Conference on Artificial Intelligence*. AAAI Press, 1711-1717 (2016)
16. Zhao F, Huang Y, Wang L, et al. J. *Computer Vision & Pattern Recognition*. 1556-1564 (2015)
17. Zhang R, Lin L, Zhang R, et al. J. *IEEE Transactions on Image Processing*, 24(12):4766-4779 (2015)
18. Lai H, Pan Y, Liu Y, et al. C. *IEEE Conference on Computer Vision and Pattern Recognition*. 3270-3278 (2015)