LETTER

Unsupervised Deep Embedded Hashing for Large-Scale Image Retrieval*

Huanmin WANG[†a)], ***Member***

SUMMARY Hashing methods have proven to be effective algorithm for image retrieval. However, learning discriminative hash codes is challenging for unsupervised models. In this paper, we propose a novel distinguishable image retrieval framework, named Unsupervised Deep Embedded Hashing (UDEH), to recursively learn discriminative clustering through soft clustering models and generate highly similar binary codes. We reduce the data dimension by auto-encoder and apply binary constraint loss to reduce quantization error. UDEH can be jointly optimized by standard stochastic gradient descent (SGD) in the embedd layer. We conducted a comprehensive experiment on two popular datasets.

key words: *hashing, unsupervised learning, deep learning*

1. Introduction

Recently, in order to solve the problem of retrieval in large-scale images [19], the computer vision community has focused on hashing algorithm which learns similarity-preserving binary codes. Encoding a high-dimensional image descriptor into a compact binary codes can improve storage efficiency and computational efficiency of similar searches, and can be implemented using data structures and algorithms that are simpler than other large-scale retrieval methods. The existing unsupervised hashing algorithms can be simply classified into two main types according to whether similarity information is precomputed before training. The first class usually directly employs deep neural networks to generate binary codes with some external constraints. For example, Deep Hashing (DH) [2] and Deepbit [3] exploit evenly distribution loss and the quantization loss to optimize the generated binary codes, which can only preserve the variance of binary codes but cannot obtain the real distribution of the raw data. To preserve more information through deep models without pre-computed similarities, some researchers leverage self-supervised algorithms such as reconstruction [4] or discriminator [5], which can help to improve the retrieval performance. However, they still have the common issues as DH and Deepbit and the performance is limited.

Methods of the second type try to conserve the similarities of the original data. Anchor Graph Hashing (AGH) [6] utilize the dense anchor map to obtain low rank adjacency matrices and discover neighborhoods in the training data, but AGH contains all the similarity values of the whole datasets and some of them might be redundant, which often leads to performance degradation. Pseudo label based methods [7], [8] convert the unsupervised learning into supervised manner by generating pseudo labels for training data. They apply clustering methods to generate pseudo labels, which can avoid useless similarity as they only preserve part of such information before training. However, most of them are sensitive to the pre-defined class number [9], which greatly limits their applications. Pseudo pair based methods generate similar and dissimilar pairs to avoid the problem of class number setting [10], [11], but it still need to define the threshold of similarity for pair generation. Besides, these methods usually compute the pseudo labels or pseudo pairs in the high-dimensional visual feature space, which is computationally complex.

In order to solve the above problem, in this paper, we introduce a novel end-to-end deep hashing algorithm for image retrieval, namely Unsupervised Deep Embedded Hashing (UDEH), which is capable of iteratively learn to cluster in the network and yield binary codes with preserving the structures of the input data distributions. Our contributions can be summarized as follows: (1) We propose a novel hashing framework that imposes constraints on the clustering space. The generated binary code has spatial structure protection; (2) Our UDEH is an end-to-end deep network where all objective functions can be optimized together by using a stochastic gradient decent (SGD) with mini batches, and our algorithm does not rely on pre-calculation.; (3) We conduct experiments on two challenging datasets (i.e., CIFAR-10 [1], NUS-WIDE [12]), and the results show that our UDEH consumes very short runtime and has superior Mean Average Precision (MAP).

2. Unsupervised Deep Embedded Hashing Method

Our task is to process a set of training samples $X = \{x_1, x_2, \cdots, x_i, \cdots, x_n\} \in \mathbb{R}^{d \times n}$, without labels, where d is the dimension of the image sample and n is the number of image samples. The purposed algorithm is to learn a set of compact binary codes $B = \{b_1, b_2, \cdots, b_i, \cdots, b_n\} \in \{-1, +1\}^{r \times n}$, which preserves the semantic similarity of the original data. Therefore, we need a valid hash function $\mathcal{H} : x \rightarrow b$ to map the data to the low-dimensional Hamming space.

In order to learn the discriminative hash code, we assign

Manuscript received May 19, 2020.
Manuscript revised June 15, 2020.
Manuscript publicized July 14, 2020.

[†]The author is with the Mechanical and Electronic Technology Institute of Lanzhou Jiaotong University, China.

[*]This work was supported by the National Natural Science Foundation of China (No. 61563027).

a) E-mail: lzzyrc@163.com

DOI: 10.1587/transfun.2020EAL2056

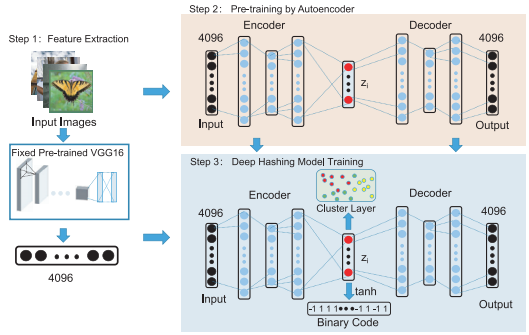

Fig. 1 An illustration of the proposed framework.

the hash layer after the reduced dimension to the k cluster centers. To solve this problem, Unsupervised Deep Embedded Hashing (UDEH) is designed to learn nonlinear hash functions with the following characteristics: t-distribution and binary quantization of learning image data. The entire learning framework consists of a feature extraction module, a pre-training module, and a model training module, as shown in Fig. 1. In the feature extraction module, we will generate a 4096 dimensional feature representation of the VGG-16 [13] model, Which has been pre-trained by imagenet. The parameters of the pre-training model are then initialized using an auto-encoder consisting of four encoder layers and an equal number of decoder layers. The parameters of the pre-training model will then be transferred for use in a hashing model with the same auto-encoder structure. After performing the automatic encoder pre-training, we obtain an initial estimate of the encoder's nonlinear map g_θ. Inspired by DEC [14], we introduced the auxiliary distribution μ_j, which is initialized by the k-means algorithm. Students t-distribution like DEC is used to measure code point z_i and centroid μ_j:

$$q_{ij} = \frac{(1+\|z_i - \mu_j\|^2)^{-1}}{\sum_{j'}(1+\|z_i - \mu_{j'}\|^2)^{-1}}, \tag{1}$$

We use the k-meas algorithm to initialize the centroid μ_j and iteratively optimize it like DEC [14]. In order to train the neural network, we need to introduce the target distribution instead of the ground truth. The target distribution can be represented as:

$$p_{ij} = \frac{q_{ij}^2/s_j}{\sum_\ell q_{i\ell}^2/s_\ell}, \tag{2}$$

where, $s_j = \sum_i q_{ij}$ is the soft cluster frequency. Since it is known that q_{ij} represents for the probability of assigning data point i to class j, it should fall in the interval of $[0, 1]$. The unknown ground truth for q_{ij} is 0/1, thus, in order to bring the results closer to 0/1, an effective way is to square q_{ij} and normalize it. After the process of Eq. (2), p_{ij} is much closer to 0/1 than q_{ij}, thence it can satisfy the first two properties. In addition, p_{ij} is also the probability, so it should fall in the interval of $[0, 1]$ too. Thus, normalization in Eq. (2) can enhance the data assignment to centroid with high confidence, and normalize loss contribution of each centroid to avoid over clustering in hidden code space. KL divergence is used to measure the similarity between two probability distributions, our effective loss function with KL divergence can be written as follows:

$$\mathcal{L}_c = -\frac{1}{n}\sum_i^n \sum_{j=1}^k p_{ij} \log \frac{p_{ij}}{q_{ij}}; \tag{3}$$

As our UDEH is a data-dependent approach, the clustering is required finding out the most suitable clusters while preserving the spatial structures of the raw data distribution. The embedded point z_i also needs to be iterative for the clustering. To overcome this challenge, we exploit a decoder to reconstruct the input feature x_i, where the reconstruction loss is measured by Mean Squared Error (MSE):

$$\mathcal{L}_r = \frac{1}{n}\sum_i^n \|f_i - g'_{\theta'}(g_\theta(f_i))\|^2, \tag{4}$$

where, g and g' are encoder and decoder respectively, f_i represents the 4096-dimensional VGG-16 feature of the sample and θ and θ' are their corresponding network parameters.

3. Binary Quantization

In order to get the binary code, it is necessary to reduce the quantization error. Specifically, we use the $tanh$ activation function to constrain the intermediate layer z_i such that the value of z_i is in the interval $[-1, 1]$. In addition, to further reduce the quantization error, we introduce the following loss function,

$$\begin{aligned} \mathcal{L}_b =& \frac{1}{n}\sum_i^n \|z_i - b_i\|^2, \\ s.t. \quad & b_i \in \{-1, 1\}^\ell \end{aligned} \tag{5}$$

where, $b_i = sign(z_i)$. The final loss function is defined as:

$$\begin{aligned} \mathcal{L} =& \mathcal{L}_c + \lambda_1 \mathcal{L}_r + \lambda_2 \mathcal{L}_b \\ =& -\frac{1}{n}\sum_i^n \sum_{j=1}^k p_{ij} \log \frac{p_{ij}}{q_{ij}} + \lambda_1 \|f_i - g'_{\theta'}(g_\theta(f_i))\|^2 \\ & + \lambda_2 \|z_i - b_i\|^2, \end{aligned} \tag{6}$$

4. Optimization

We fixed the target distribution p_{ij} and then calculated the gradient of L_c relative to each projection data point z_i and each cluster center μ_j as:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial z_i} =& 2\sum_{j=1}^k (1 + \|z_i - \mu_j\|^2)^{-1} \times (p_{ij} - q_{ij})(z_i - \mu_j) \\ & + \frac{2\lambda_1}{n}(f_i - g'_{\theta'}(z_i))\frac{\partial g'_{\theta'}(z_i)}{\partial z_i} + \frac{2\lambda_2}{n}(z_i - b_i), \end{aligned} \tag{7}$$

and each cluster center μ_j are computed as,

$$\frac{\partial \mathcal{L}}{\partial \mu_j} = -2 \sum_{i=1}^{n} (1 + \|z_i - \mu_j\|^2)^{-1} \times (p_{ij} - q_{ij})(z_i - \mu_j). \tag{8}$$

The above gradient $\frac{\partial \mathcal{L}}{\partial z_i}$ is passed down to the neural network and used to calculate its parameters in standard Back-Propagation (BP) Gradient $\frac{\partial \mathcal{L}}{\partial \theta}$, then θ can be used with $\theta = \theta - \eta_1 \frac{\partial \mathcal{L}}{\partial \theta}$, where η_1 is the learning rate.

5. Performance Analysis

Evaluation Metrics: The indicator for evaluating these algorithms is mean average precision (mAP). Suppose the returned P images has p right results, mAP can be defined as,

$$mAP = \frac{1}{p} \sum_{i=1}^{p} \frac{i}{P_i} \tag{9}$$

where, i is the i-th correct retrieved image, and P_i stands for the position of the i-th correct retrieved image in the returned queue.

We evaluate our UDTH and the baselines on three different lengths of binary code $\{16, 32, 64\}$. For the cifar dataset, we compute the mAP value from the top 1000 retrieved samples. On NUS-Wide and MIRFLICKR-25k, the number is set as 5000. Besides, the Precision and Recall are also widely used in the field of classification and retrieval to express the performance, their definitions are,

$$Precision = \frac{p}{P}, \quad Recall = \frac{p}{Q}, \tag{10}$$

where, Q is the total number of the image which share the same label with the query image in the dataset.

Results: Before discussing the experimental results, we first declare that all comparison experiments have the same input, that is, 4096-dimensional VGG-16 features. On CIFAR-10, we randomly pick 100 images from each category for training and the left 5,900 images are included in test set. On NUS-WIDE, 2,100 images are randomly selected from the dataset for test and the remaining 157497 images are left for training. Leaing rate is 0.0001, $\lambda_1 = 0.1$ and $\lambda_2 = 0.1$, After learning the fixed-length binary code, our model enters the query image and returns the image from the training set according to the Hamming distance. Table 1 shows the performance of UDEH. We can see that the experimental results of UDEH on all bits are very good. In addition, we can see that as the bit size grows, the value of mAP increases steadily, so we can infer the algorithm to perform better on larger bit sizes. We also compare UDEH-nr (UDEH without $\mathcal{L}_r$), the performance of UDEH-nr is slightly lower. Table 2 shows the advantages of our algorithm in time complexity, including Training time, Generation time and Search time. From the data in the table, it is not difficult to find that the

Table 1 The following table shows the retrieval performance of 64-bit codes. (MAP@1,000 on CIFAR-10, and MAP@5,000 on NUS-WIDE).

Method	CIFAR-10			NUS-WIDE		
	16	32	64	16	32	64
ITQ	0.311	0.328	0.347	0.511	0.516	0.524
PCAH	0.212	0.186	0.168	0.410	0.392	0.376
LSH	0.179	0.214	0.217	0.410	0.406	0.438
DSH	0.246	0.262	0.292	0.501	0.491	0.518
SpH	0.204	0.237	0.259	0.418	0.455	0.473
SELVE	0.309	0.280	0.239	0.467	0.462	0.432
UN-BDNN	0.264	0.281	0.294	0.470	0.472	0.481
Deepbit	0.207	0.209	0.244	0.409	0.409	0.439
DDH	0.283	0.301	0.312	0.472	0.475	0.489
UDEH-nr	0.369	0.415	0.437	0.594	0.599	0.620
UDEH (ours)	**0.383**	**0.428**	**0.455**	**0.602**	**0.625**	**0.641**

Table 2 The following table shows that our method consumes very little time, and unlike the pseudo-label algorithm, no pre-calculation is required.

Dataset	CIFAR-10	NUS-WIDE
MAP	0.5673	0.7357
Training time (s)	561	1195
Generation time (s)	3.68	8.42
Search time (ms)	2.71	3.02

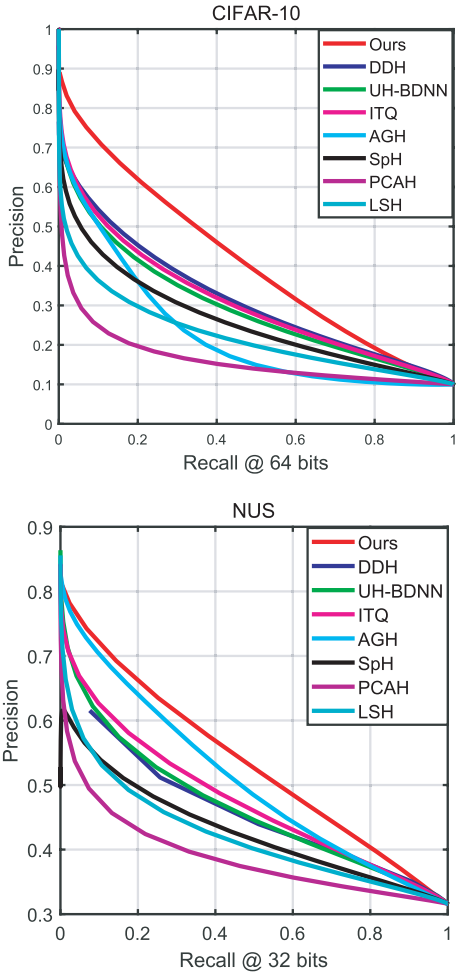

Fig. 2 The precision recall (PR) curve of CIFAR-10 and NUS-WIDE.

algorithm in this paper consumes very little time to train and generate code, and does not require pre-training. Figure 2 shows the Precision Recall (PR) curve of the hash code on the CIFAR-10 and NUS-WIDE datasets. In order to evaluate the effectiveness of the proposed method, we compare the results of the proposed method and the comparison method, including DDH [10], UH-BDNN [4], ITQ [15], AGH [6], SpH [16], PCAH [17], LSH [18]. The length of the hash code in the CIFAR-10 experiment is 64 bits, and the NUS-WIDE experiment is 32 bits. From the curve we can see that the return image with the shortest distance from the query image Hamming has a high accuracy. The curve of our method is much higher than all other comparison methods, which proves the effectiveness of this algorithm.

6. Conclusion

Our proposed Unsupervised Deep Embedded Hashing (UDEH) performs clustering and quantification tasks simultaneously in a low-dimensional space, and retains the structural information of the raw image with reconstruction loss. Compared with the previous image retrieval methods, UDEH has great advantages in terms of time complexity and retrieval precision.

References

[1] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Technical Report, Citeseer, 2009.
[2] V.E. Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou, "Deep hashing for compact binary codes learning," IEEE Conference on Computer Vision and Pattern Recognition, pp.2475–2483, 2015.
[3] K. Lin, J. Lu, C. Chen, and J. Zhou, "Learning compact binary descriptors with unsupervised deep neural networks," IEEE Conference on Computer Vision and Pattern Recognition, pp.1183–1192, 2016.
[4] T.T. Do, A.D. Doan, and N.M. Cheung, "Learning to hash with binary deep neural network," European Conference on Computer Vision, pp.219–234, 2016.
[5] K. Ghasedi Dizaji, F. Zheng, N. Sadoughi, Y. Yang, C. Deng, and H. Huang, "Unsupervised deep generative adversarial hashing network," IEEE Conference on Computer Vision and Pattern Recognition, pp.3664–3673, 2018.
[6] W. Liu, J. Wang, S. Kumar, and S.-F. Chang, "Hashing with graphs," International Conference on Machine Learning, pp.1–8, 2011.
[7] Q. Hu, J. Wu, J. Cheng, L. Wu, and H. Lu, "Pseudo label based unsupervised deep discriminative hashing for image retrieval," ACM Conference on Multimedia, pp.1584–1590, 2017.
[8] J. Song, L. Gao, Y. Yan, D. Zhang, and N. Sebe, "Supervised hashing with pseudo labels for scalable multimedia retrieval," ACM Conference on Multimedia, pp.827–830, 2015.
[9] L. Xie, L. Zhu, P. Pan, and Y. Lu, "Cross-modal self-taught hashing for large-scale image retrieval," Signal Process., vol.124, pp.81–92, 2016.
[10] J. Song, T. He, H. Fan, and L. Gao, "Deep discrete hashing with self-supervised pairwise labels," The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, pp.223–238, 2017.
[11] E. Yang, C. Deng, T. Liu, W. Liu, and D. Tao, "Semantic structure-based unsupervised deep hashing," International Joint Conferences on Artificial Intelligence, pp.1064–1070, 2018.
[12] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng, "NUS-WIDE: A real-world web image database from national university of singapore," International Conference on Image and Video Retrieval, 2009.
[13] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," International Conference on Learning Representations, pp.1–29, 2015.
[14] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," International Conference on Machine Learning, pp.478–487, 2016.
[15] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," IEEE Trans. Pattern Anal. Mach. Intell., vol.35, no.12, pp.2916–2929, 2013.
[16] J.P. Heo, Y. Lee, J. He, S.F. Chang, and S.E. Yoon, "Spherical hashing," IEEE Conference on Computer Vision and Pattern Recognition, pp.2957–2964, 2012.
[17] J. Wang, S. Kumar, and S.-F. Chang, "Semi-supervised hashing for large-scale search," IEEE Trans. Pattern Anal. Mach. Intell., vol.35, no.12, pp.2393–2406, 2012.
[18] M.S. Charikar, "Similarity estimation techniques from rounding algorithms," International Conference on Image and Video Retrieval, pp.380–388, 2002.
[19] J. Chen, W.K. Cheung, and A. Wang, "Learning deep unsupervised binary codes for image retrieval," International Joint Conference on Artificial Intelligence, pp.613–619, 2018.