



Deep Supervised Hashing by Classification for Image Retrieval

Xiao Luo^{1,2}, Yuhang Guo¹, Zeyu Ma³, Huasong Zhong², Tao Li¹, Wei Ju⁴,
Chong Chen², and Minghua Deng¹(✉)

¹ School of Mathematical Sciences, Peking University, Beijing, China
{xiaoluo,yuhangguo,li_tao,dengmh}@pku.edu.cn

² Damo Academy, Alibaba Group, Hangzhou, China
{huasong.zhs,cheung.cc}@alibaba-inc.com

³ School of Computer Science and Technology, Harbin Institute of Technology,
Shenzhen, China
zeyu.ma@stu.hit.edu.cn

⁴ Department of Computer Science and Technology, Peking University,
Beijing, China
juwei@pku.edu.cn

Abstract. Hashing has been widely used to approximate the nearest neighbor search for image retrieval due to its high computation efficiency and low storage requirement. With the development of deep learning, a series of deep supervised methods were proposed for end-to-end binary code learning. However, the similarity between each pair of images is simply defined by whether they belong to the same class or contain common objects, which ignores the heterogeneity within the class. Therefore, those existing methods have not fully addressed the problem and their results are far from satisfactory. Besides, it is difficult and impractical to apply those methods to large-scale datasets. In this paper, we propose a brand new perspective to look into the nature of deep supervised hashing and show that classification models can be directly utilized to generate hashing codes. We also provide a new deep hashing architecture called **Deep Supervised Hashing by Classification (DSHC)** which takes advantage of both inter-class and intra-class heterogeneity. Experiments on benchmark datasets show that our method outperforms the state-of-the-art supervised hashing methods on accuracy and efficiency.

Keywords: Learning to hash · Supervised learning · Deep hashing

1 Introduction

In recent years, hundreds of thousands of images are generated in the real world every day, making it extremely difficult to find the relevant images. Due to the effectiveness of deep convolution neural networks, images either in the

X. Luo, Y. Guo and Z. Ma—Contribute equally to this work. The work was done when Xiao Luo interned in Damo Academy, Alibaba Group.

© Springer Nature Switzerland AG 2021

T. Mantoro et al. (Eds.): ICONIP 2021, LNCS 13111, pp. 3–14, 2021.

https://doi.org/10.1007/978-3-030-92273-3_1

database or the query image can be well represented by real-valued features. Therefore image retrieval can be addressed as an approximating nearest neighbor (ANN) searching problem for the sake of computational efficiency and high retrieval quantity. Compared to the traditional content-based methods, hashing methods has shown its superiority for data compression, which transforms high-dimensional media data into the generated binary representation [4, 11]. There are a number of learning-to-hashing methods for efficient ANN searching [14], which mainly fall into unsupervised methods [4, 13] and supervised methods [11, 12]. As the development of deep learning, deep hashing methods have prevailed and shown competitive performance for their ability to learn image representation [7]. By transferring deep representation learned by deep neural networks, effective hash codes are obtained by controlling the loss function. Specifically, they can learn similar-preserving representations and control quantization error for continuous representation by converting into binary codes. These methods can also be divided into three schemes, pairwise label based methods [1, 9], triplet label based methods [24] and point-wise classification schemes [10, 21], respectively. It's noticed that the above schemes can be mixed and utilized together [8]. Recently, several methods have added label information into their models and achieved great success [8].

Although these existing methods have achieved considerable progress, two significant drawbacks have not been fully addressed yet. The supervised hashing methods are usually guided by a similarity matrix S , while the definition of S is quite simple. Specifically, $s_{ij} = 1$ if image i and image j belong to the same class or contain common objects, and $s_{ij} = 0$ otherwise. Definitely, this way of definition is reasonable in a sense since images of the same category are considered to be the same. However, there are usually many sub-classes in the same class, and there should be some differences between different sub-classes. If all the sub-classes are forced to be regarded as the same, the obtained hash codes will be very unstable [15], so that the extension results on the test set will be poor. Therefore, the existing methods do not fully proceed from the perspective of image retrieval, and thus leading to unsatisfied retrieval accuracy. On the other hand, we notice that the schemes mentioned above often include complex pairwise loss functions, which means training on large datasets is difficult. Therefore, VGG-Net [16] is often used in deep supervised hashing task for the sake of speeding. If deeper models like ResNet are used [5], we need to replace the loss function of deep hashing with simpler forms such as those in the classification problem.

To address the two disadvantages mentioned above, we investigate the relationship between deep supervised hashing and classification problems. It turns out that high-quality binary codes can be generated by deep classification models. For single-label datasets, we can construct the mapping relationship between the classification result and the hash code in some ways, making the hamming distance between different classes of hash codes relatively large, while the hamming distance between the subclasses of the same class relatively small. For multi-label datasets, it is very natural to regard the predicted multi-shot labels as the final hash codes, since the dissimilarity of two images is well captured

by the hamming distance. Following this idea, we proposed Deep Supervised Hashing by Classification (DSHC), a novel deep hashing model that can generate effective and concentrated hash codes to enable effective and efficient image retrieval. The main contributions of DSHC are outlined as follows:

- DSHC is an end-to-end hash codes generation framework containing three main components: 1) a standard deep convolutional neural network (CNN) such as ResNet101 or ResNeXt, 2) a novel classification loss based on cross-entropy that helps to divide the origin classed into several sub-classes by their features and 3) a heuristic mapping from sub-labels to hash codes making the hash codes of the sub-classed with the same label approach in Hamming space.
- To the best of our knowledge, DSHC is the first method that addresses deep supervised hashing as a classification problem and looks into the heterogeneity within the class.
- Comprehensive empirical evidence and analysis show that the proposed DSHC can generate compact binary codes and obtain state-of-the-art results on both CIFAR-10 and NUS-WIDE image retrieval benchmarks.

2 Related Work

Existing hashing methods can be divided into two categories: unsupervised hashing and supervised hashing methods. We can refer to [14] for a comprehensive survey. In unsupervised hashing methods, data points are encoded to binary codes by training from unlabeled data. Typical methods are based on reconstruction error minimization or graph learning [4, 19]. Supervised hashing further makes use of supervised information such as pair-wise similarity and label information to generate compact and discriminative hash codes. Across similar pairs of data points, nonlinear or discrete binary hash codes are generated by minimizing the Hamming distances and vice versa across dissimilar pairs [11, 15].

Deep hash learning demonstrates their superiority over shallow learning methods in the field of image retrieval through powerful representations. Specifically, Deep Supervised Discrete Hashing [8] combines CNN model with a probability model to preserve pairwise similarity and regress labels using hash codes simultaneously. Deep Hashing Network [25] proposes the first end-to-end framework which jointly preserves pairwise similarity and reduces the quantization error between data points and binary codes. To satisfy Hamming space retrieval, DPSH [9] introduces a novel cross-entropy loss for similarity-preserving and the quantization loss for controlling hashing quality. However, these methods are difficult to apply to large-scale datasets that ignore heterogeneity within classes. Recently, [22] uses a self-learning strategy and improves the performance.

Image Classification Tasks including single-label and multi-label have been made impressive progress by using deep convolutional neural networks. Single-label image classification, paying attention to assign a label from a predefined set to an image, has been extensively studied. The performance of single-label image classification has surpassed human in ImageNet dataset [5]. However,

multi-label image classification is a more practical and general problem, because the majority of images in the real-world contain more than one object from different categories.

A simple and straightforward way for multi-label image classification is training an independent binary classifier for each class. However, this method does not consider the relationship among classes! Indeed, the relationship between different labels can be considered by graph neural networks [2]. Additionally, Wang et al. [17] adopted recurrent neural network (RNN) to encode labels into embedded label vectors, which can employ the correlation between labels. Recently, the attention mechanism and label balancing have been introduced to discover the label correlation for multi-label image classification. Different from other hash methods, this paper treats the generation of hash codes as a classification task.

3 Approach

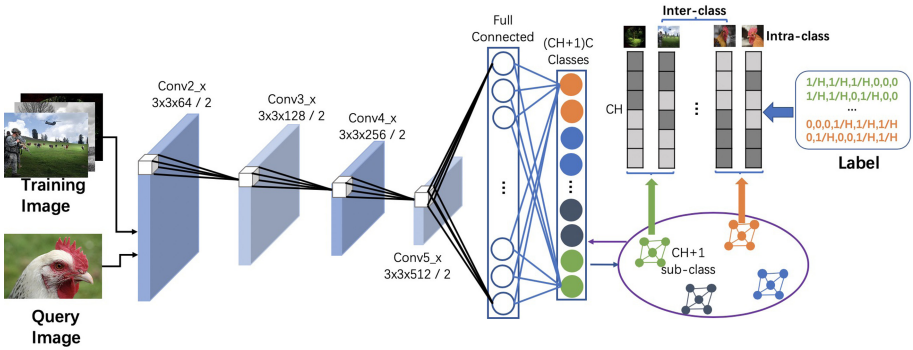


Fig. 1. Overview of our proposed method: The CNN layers perform feature extracting followed by fully connected layer with soft-max to output $(CH + 1)C$ sub-classes. There are $(CH + 1)$ sub-class contained for each class. And each sub-class will be mapped into a hash code with length CH . The multi-hot labels are employed to optimize the network. (Best viewed in color) (Color figure online)

3.1 Problem Formulation

In the problem of image retrieval, given a dataset $\mathcal{O} = \{\mathbf{o}_i\}_{i=1}^n$, $\mathbf{o}_i = (\mathbf{x}_i, \mathbf{l}_i)$, in which \mathbf{x}_i is the feature of the i -th image, and $\mathbf{l}_i = [l_{i1}, \dots, l_{iC}]$ is the label annotation assigned to the i -th image, in which C is the number of classes. The similarity label $s_{ij} = 1$ implies the i -th image and the j -th image are similar, otherwise $s_{ij} = 0$. The similar pairs are constructed by the image labels, i.e. two images will be considered similar if they have at least one common label. The goal of deep hashing is to learn a non-linear hash function: $f : \mathbf{o} \rightarrow \mathbf{h} \in \{-1, 1\}^L$, encoding each sample \mathbf{o} into compact L -bit hash code \mathbf{h} where original similarities between sample pairs are well preserved. For computational

consideration, the distance between different hash codes is Hamming distance, which can be formulated as

$$\text{dis}_H(\mathbf{h}_i, \mathbf{h}_j) = \frac{1}{2} (L - \langle \mathbf{h}_i, \mathbf{h}_j \rangle)$$

where \langle, \rangle denotes the inner product of hash codes.

3.2 Mapping Sub-classes to Hash Codes

As mentioned in the introduction, we would like to construct a mapping from sub-classes to hash codes, such that the hamming distances in the same class are relatively smaller than those between classes. Suppose we have C classes and each class can be divided into m sub-classes. And for j -th sub-class of i -th class, it has a unique hash code mapping p_{ij} . Then we define

$$D_{inter} = \sum_{i_1} \sum_{i_2 \neq i_1} \left(\sum_{j_1} \sum_{j_2} \text{dist}_H(p_{i_1 j_1}, p_{i_2 j_2}) \right)$$

$$D_{intra} = \sum_i \sum_{j_1} \sum_{j_2} \text{dist}_H(p_{i j_1}, p_{i j_2})$$

as the total inter-class and intra-class distances respectively. Given the code length L , we aim to find $C \times m$ hash codes such that

$$D_{intra} - D_{inter} \tag{1}$$

is minimized. However, finding the global optimization of the objective function 1 is NP-hard, so we proposed a space partition based method to get an approximate solution of it.

As shown in Fig. 1, suppose the hamming space with dimension L is well separated into m sub-spaces, each sub-space corresponding to a class. For each subspace i , suppose there is a center point p_i , which can be viewed as the benchmark code of class i . Then for each subclass of class i , we just substitute one position of p_i , thus every sub-class is mapped to a unique hash code eventually. It is easy to check that all hamming distances in the same class are smaller than or equal to 2. So we can get high-quality hash codes if the hamming distances between center points are much larger than 2.

The most critical step is to construct the center points that are well separated. To make this purpose, we proposed two methods named Center Point-Based Construction (CPBC) and K-means Based Construction (KBC). The idea of CPBC is simple but effective. Specifically, assume each class has a sub-hash code with a length of H , and the total code length is CH . For a center point p_i , the i -th sub-hash code (length H) is all set to be 1 and other sub-hash codes are -1 . Then all the center points can be determined, while the Hamming distance between any two of them is equal to $2H$. We can see that hash codes generated by CPBC are generally well separated through T-SNE clustering (see Fig. 2). For CPBC, we have to choose a relatively big H to get high-quality center

points. KBC is proposed to generate relatively shorter but high-quality center points. First, we choose C initial points with a given hash code length L , and all the 2^L points are clustered into C groups by K-means. Then the resulted in C clustering centroids are viewed as C center points. Since each cluster contains many more points than the number of subclasses, all subclasses are only mapping to a small ball centered in the corresponding center point. Theoretically, KPC needs a shorter hash code than CPBC, but we found that the effect is difficult to guarantee in the numeric experiments. So we always use CPBC when comparing with other methods.

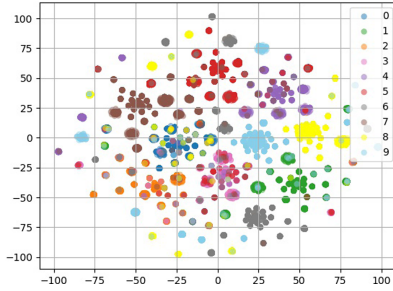


Fig. 2. T-SNE clustering visualization of hash codes generated by CPBC with CIFAR10 data set. Axes represent the first two dimensions of t-SNE embedding.

3.3 Loss Function

Our model converts learning to hash into a classification problem by introducing multiple subclasses for each superclass. To learn a novel function that maps each subclass to a unique hash code, all we need is to determine which subclass each image belongs to. In this section, we will see that the first loss is the cross-entropy loss for a single-label dataset, while the second loss is the binary cross-entropy and soft-margin loss for a multi-label dataset.

For the single-label image classification, the cross-entropy loss is usually employed [5]. Suppose there are C classes in the training dataset and sub-hash code with length H for each class, the whole length of the hash code is CH . In intra-class, one of the hash codes can be replaced to generate CH sub-classes. Thus $(CH + 1)C$ classes will be contained in the soft-max output. Correspondingly, the one-hot ground truth will be converted into the multi-hot, in which the $(CH + 1)$ label points are assigned to $1/H$ or otherwise set to 0. Here we use $1/H$ instead of $1/(CH + 1)$ for computation. Formally, the loss function is:

$$L_{ce} = - \sum_{c=1}^{(CH+1)C} y_c \log(p_c), \quad (2)$$

where $y_c \in \{0, 1/H\}$ is the ground truth and p_c is the predicted probability distribution. Due to considering the robustness of the model, the top-K prediction

probabilities are chosen to generate the corresponding hash codes. And the final hash code can be calculated by the average of K hash codes.

As for the multi-label image classification, the labels are multi-hot values which are contained in the ground truth so that they can be utilized to express the hash code directly. There are relatively large Hamming distances between intra-class and relatively small Hamming distances within inter-class. Besides, our method can explore the correlation between classes rather than predict correctly as long as one label is matched. Since the loss with soft margin is introduced to address the multi-label classification task, the loss is computed as:

$$L_{x,y} = - \sum_{i=1}^N y_i \log((1 + \exp(-x_i))^{-1}) + (1 - y_i) \log((1 + \exp(x_i))^{-1}), \quad (3)$$

where $y_i \in \{1, -1\}$ express positive or negative class. x_i and N are the predicted probability and the batch size of data in the training phase, respectively.

4 Experiments

The performance of our proposed approach is evaluated on two public benchmark datasets: CIFAR-10 [6] and NUS-WIDE [3] comparing with state-of-the-art methods.

4.1 Datasets and Settings

CIFAR-10. CIFAR-10 is a dataset containing 10 object categories, each with 6000 images (resolution of 32×32). We sampled 1,000 images per class (10,000 images in total) as the query set and the remaining 50,000 images were utilized as the training set and database for retrieval.

NUS-WIDE. NUS-WIDE is a public multi-label image dataset consisting of 269,648 images. Each image is manually annotated using some of the 81 ground truth concepts for evaluating retrieval models. Following [8], we picked a subset of 195,834 images associated with 21 most frequent labels. We randomly sampled 2,100 images as query sets and the remaining images were treated as the training set.

The retrieval quality is evaluated by the following four evaluation metrics: Mean Average Precision(MAP), Precision-Recall curves, Precision curves concerning hamming radius, and Recall curves for hamming radius. We measure the goodness of the result by comprehensively calculating MAP. For NUS-WIDE, for each bit, the distance needs to be different when the values are all 1s or all 0s when calculating the distance between two images. As a result, we convert -1 to 0 in hash codes and the distance between two images is still in the form of Equation in Sect 3.1.

Our methods are compared with a list of classical or state-of-the-art supervised methods, including DSDH [8], DPSH [9], VDSH [23], DTSH [18], RMLH

[20] and unsupervised hashing methods including SH [19], ITQ [4]. For CIFAR10, we utilize ResNet50 and replace the last layers with the corresponding number of nodes, with the learning rate 0.1. We also rerun the source code of DPSH and DSDH. The number of total epochs is 160 since we found all models can fit very well afterward. For NUS-WIDE, we utilize ResNet 101 and the learning rate is set to 0.1 which decreases every 6 epochs.

4.2 Performance

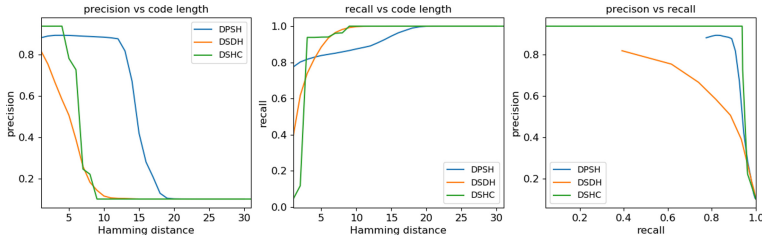


Fig. 3. Precision curves, Recall curves respect to hamming radius and Precision-Recall curves when code length is 30. In the first two figures, the x axis represents the Hamming distance (radius), and the y axis represents the average precision and recall. The last figure is the curve of precision and recall.

Table 1 shows the results of different hashing methods on two benchmark data sets when the code length is about 32 and 24, respectively. Here, for our method, the code length is a little smaller than 32, but they are comparable because the last two or three bits can be filled with zero for all images. Figure 3 and Table 1 shows the Precision-Recall curves, Precision curves, and Recall curves respectively for different methods (code length of 30 and 60 bits).

We find that on the two benchmark datasets, DSHC outperforms all the compared baseline methods. What’s more, unsupervised traditional hashing methods show poor performances, which implies that labels and the strong representation ability of deep learning are significant for learning to the hash. Compared with DSDH which regresses the labels with hash codes, our method directly utilizes the labels to produce sub-labels, showing superiority from the increment of performance. Compared with DPSH, our model is based on a deeper model such as ResNet50 for the sake of getting rid of the pairwise loss whose computation cost increases greatly. As shown in Table 2, our method takes the advantage of a deeper model like ResNet but with less training time, which implies it can easily extend to large-scale image datasets. Figure 3 shows that when code length varies from 60 to 30, the performance of our method is stable while those of DPSH and DSDH are sensitive to the code length. Besides, when the code length is 60, the average recall for the code distance 0 in our model is about 0.024 while the value of DPSH is about 0.724. In other words, most images with the same labels

Table 1. MAP for different methods on CIFAR10 dataset and NUS-WIDE dataset. The MAP for the NUS-WIDE dataset is calculated based on the top 50000 returned neighbors. We re-run the source codes of DSDH and DPSH with code length 30, shown in the brackets while other results are from their papers.

Method	Published year	CIFAR10		NUS-WIDE	
		MAP	Length(bits)	MAP	Length(bits)
DSHC	Ours	0.9431	30	0.844	21
RMLH	2019	0.816	32	0.823	32
DSDH*	2017	0.939(0.802)	32(30)	0.814	24
DPSH*	2016	0.781(0.887)	32(30)	0.722	24
DTSH	2016	0.925	32	0.776	24
VDSH	2017	0.844	32	0.564	24
ITQ	2012	0.172	32	0.468	24
SH	2009	0.126	32	0.406	24

in DPSH are projected into the same hash codes while our method can retrieval the most similar images within the class.

Table 2. MAP for different methods on CIFAR10 dataset (60-bit hash codes)

Method	MAP	Runtime (Last Ten epochs)
DSHC	0.9437	409.27
DPSH	0.8990	600.30
DSDH	0.8786	352.64

Since NUS-WIDE is annotated with multi-labels, we directly use the classification binary output as hash codes. The results show that this kind of hash code works quite well and performs much better than other methods. A reasonable explanation is that the binary classification output has already captured the intra-class heterogeneity of the dataset. What’s more, the multi-labeled classification model considers the relationship between sub-labels while most deep hash methods only consider the similarity between the two images.

4.3 Results with Different Code Length

We also compare the performance of our model with different code lengths. Taking the CIFAR10 dataset, for instance, the result is shown in Table 3. When a short hash code is used, CPBC is not able to partition the space well, resulting in poor performance. When the hash code length is large enough (e.g. above 30), the MAP of our model is quite stable. Under the condition of a complex

dataset, more hash bits will be needed for acceptable performance. It is worth noting that the length of the hash code does not influence the running time.

Table 3. MAP and Runtime (Last Ten epochs) for different length

Length	MAP	Runtime
10	0.3599	408.21
20	0.8264	427.09
30	0.9431	420.92
60	0.9437	409.27

4.4 Comparison Between CPBC and KBC

As mentioned above, the performance of our method is difficult to guarantee when the hash code length is small (e.g., 10). However, if KBC is used to obtain the centroids, its performance is acceptable even if the hash code length is as small as 10, while CPBC is difficult to separate the Hamming space well. As shown in Table 4, when the hash code length is set to 10, 950 points out of 1024 points in the KBC model are filtered, while the model using CPBC leaves 110 points. The model using KBC performs much better than CPBC, which means that the model using KBC can successfully select several groups with larger distances in Hamming space. However, when the hash code length is large enough, it is recommended to use the CPBC model for simplification.

Table 4. MAP@5000 for CPBC and KBC methods

Length	Method	MAP@5000	Number of sub-classes
10	CPBC	0.5238	110
10	KBC	0.8980	74(filter 950)

5 Discussion

From the results, our classification method shows superior performance compared to state-of-the-art methods when the labels of the images are known. Also, it can handle large-scale datasets without dealing with pairwise losses, thus speeding up the computation. More importantly, when the classification output is directly transformed into hash codes in NUS-WIDE, this suggests that the classification model may be the key to deep supervised hashing.

Sometimes, similarity information is the only supervised information. We have two methods to obtain the labels of images. First, if the real model is simple, we can find images that contain only one label and get the exact specific label from the similarity. Second, if the real model is complex, we can construct

a similarity graph based on the information and then derive the final labels by graph clustering (e.g., Markov clustering). From the clustering results, we can derive the label for each image. However, the results are limited by the performance of clustering, which is difficult to promise.

6 Conclusion

In this paper, we investigate the relationship between deep supervised hashing and classification problems and find that high-quality hash codes can be generated by deep classification models. We propose a new supervised hashing method named DSHC, which consists of a classification module and a transformation module, and exploits inter-and intra-class heterogeneity. Based on the performance of several benchmark datasets, DSHC proves to be a promising approach. Further research can focus on designing an efficient ANN search algorithm based on hash codes generated by DSHC.

Acknowledgements. This work was supported by The National Key Research and Development Program of China (No. 2016YFA0502303) and the National Natural Science Foundation of China (No. 31871342).

References

1. Cao, Z., Long, M., Wang, J., Yu, P.S.: Hashnet: deep learning to hash by continuation. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 5608–5617 (2017)
2. Chen, T., Xu, M., Hui, X., Wu, H., Lin, L.: Learning semantic-specific graph representation for multi-label image recognition. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 522–531 (2019)
3. Chua, T.S., Tang, J., Hong, R., Li, H., Luo, Z., Zheng, Y.: Nus-wide: a real-world web image database from national university of Singapore. In: Proceedings of the ACM International Conference on Image and Video Retrieval, p. 48. ACM (2009)
4. Gong, Y., Lazebnik, S., Gordo, A., Perronnin, F.: Iterative quantization: a procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.* **35**(12), 2916–2929 (2012)
5. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
6. Krizhevsky, A., et al.: Learning multiple layers of features from tiny images (2009)
7. Lai, H., Pan, Y., Liu, Y., Yan, S.: Simultaneous feature learning and hash coding with deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3270–3278 (2015)
8. Li, Q., Sun, Z., He, R., Tan, T.: Deep supervised discrete hashing. In: Advances in Neural Information Processing Systems, pp. 2482–2491 (2017)
9. Li, W.J., Wang, S., Kang, W.C.: Feature learning based deep supervised hashing with pairwise labels. In: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, pp. 1711–1717. AAAI Press (2016)

10. Lin, K., Yang, H.F., Hsiao, J.H., Chen, C.S.: Deep learning of binary hash codes for fast image retrieval. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 27–35 (2015)
11. Liu, W., Wang, J., Ji, R., Jiang, Y.G., Chang, S.F.: Supervised hashing with kernels. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition, pp. 2074–2081. IEEE (2012)
12. Liu, X., He, J., Deng, C., Lang, B.: Collaborative hashing. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2139–2146 (2014)
13. Liu, X., He, J., Lang, B., Chang, S.F.: Hash bit selection: a unified solution for selection problems in hashing. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1570–1577 (2013)
14. Luo, X., et al.: A survey on deep hashing methods. arXiv preprint [arXiv:2003.03369](https://arxiv.org/abs/2003.03369) (2020)
15. Luo, X., et al.: Cimon: towards high-quality hash codes. In: IJCAI (2021)
16. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint [arXiv:1409.1556](https://arxiv.org/abs/1409.1556) (2014)
17. Wang, J., Yang, Y., Mao, J., Huang, Z., Huang, C., Xu, W.: CNN-RNN: a unified framework for multi-label image classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2285–2294 (2016)
18. Wang, X., Shi, Y., Kitani, K.M.: Deep supervised hashing with triplet labels. In: Lai, S.-H., Lepetit, V., Nishino, K., Sato, Y. (eds.) ACCV 2016. LNCS, vol. 10111, pp. 70–84. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-54181-5_5
19. Weiss, Y., Torralba, A., Fergus, R.: Spectral hashing. In: Advances in Neural Information Processing Systems, pp. 1753–1760 (2009)
20. Wu, L., Fang, Y., Ling, H., Chen, J., Li, P.: Robust mutual learning hashing. In: 2019 IEEE International Conference on Image Processing (ICIP), pp. 2219–2223. IEEE (2019)
21. Yang, H.F., Lin, K., Chen, C.S.: Supervised learning of semantics-preserving hashing via deep neural networks for large-scale image search. arXiv preprint [arXiv:1507.00101](https://arxiv.org/abs/1507.00101) 1(2), 3 (2015)
22. Zhan, J., Mo, Z., Zhu, Y.: Deep self-learning hashing for image retrieval. In: 2020 IEEE International Conference on Image Processing (ICIP), pp. 1556–1560. IEEE (2020)
23. Zhang, Z., Chen, Y., Saligrama, V.: Efficient training of very deep neural networks for supervised hashing. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1487–1495 (2016)
24. Zhao, F., Huang, Y., Wang, L., Tan, T.: Deep semantic ranking based hashing for multi-label image retrieval. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1556–1564 (2015)
25. Zhu, H., Long, M., Wang, J., Cao, Y.: Deep hashing network for efficient similarity retrieval. In: Thirtieth AAAI Conference on Artificial Intelligence (2016)