A Survey on Deep Hashing Methods

Xiao Luo[1,2] · **Chong Chen**[1,2*] · **Huasong Zhong**[2] · **Hao Zhang**[2] · **Minghua Deng**[1] · **Jianqiang Huang**[2] · **Xiansheng Hua**[2]

Abstract Nearest neighbor search is to find the data points in the database such that the distances from them to the query are the smallest, which is a fundamental problem in various domains, such as computer vision, recommendation systems and machine learning. Hashing is one of the most widely used method for its computational and storage efficiency. With the development of deep learning, deep hashing methods show more advantages than traditional methods. In this paper, we present a comprehensive survey of the deep hashing algorithms. Based on the loss function, we categorize deep supervised hashing methods according to the manners of preserving the similarities into: pairwise similarity preserving, multiwise similarity preserving, implicit similarity preserving, as well as quantization. In addition, we also introduce some other topics such as deep unsupervised hashing and multi-modal deep hashing methods. Meanwhile, we also present some commonly used public datasets and the scheme to measure the performance of deep hashing algorithms. Finally, we discussed some potential research directions in the conclusion.

1 Introduction

Nearest neighbor search is one of the most basic problems in many fields, such as computer vision, recommendation systems, and machine learning. Its purpose is to find the closest point from the dataset to the query based on a certain distance. However, when the amount of data is large and the dimensions are high, the time cost of accurately finding the point closest to the query is very large. To solve this problem, people began to pay more attention to approximate nearest neighbor search because in most cases it can meet the search needs and greatly reduce the search complexity.

Hashing is one of the most widely used methods because it is very efficient in terms of computation and storage. Its purpose is to convert the original features of high latitudes into low-dimensional hash codes, so that the hash codes of the original similar objects are as close as possible, and the hash codes of dissimilar objects are as different as possible. The existing hashing methods are mainly divided into two categories, one is local sensitive hashing [18,59], and the other is learning to hashing. The purpose of local sensitive hashing is to map the original data into several hash buckets, so that the closer the original distance objects are, the greater the probability that they will fall in the

Xiao Luo
E-mail: xiban.lx@alibaba-inc.com

Chong Chen
E-mail: cheung.cc@alibaba-inc.com

Huasong Zhong
E-mail: huasong.zhs@alibaba-inc.com

Hao Zhang
E-mail: puhong.zh@taobao.com

Minghua Deng
E-mail: dengmh@math.pku.edu.cn

Jianqiang Huang
E-mail: jianqiang.hjq@alibaba-inc.com

Xiansheng Hua
E-mail: xiansheng.hxs@alibaba-inc.com

1. School of Mathematical Sciences, Peking University
2. Damo Academy, Alibaba Group
*. Corresponding author

same hash bucket. Through this mechanism, many algorithms based on locally sensitive hashing have been proposed [5, 6, 27, 28, 96, 99], which show high superiority in both calculation and storage. However, in order to improve the recall rate of search, these methods usually need to build many different hash tables, so their application on particularly large data sets is still limited.

Since local sensitive hashing is data independent, people try to get high quality hashing codes by learning good hash functions. Since the two early algorithms, semantic hashing [102, 103] and spectral hashing [118] that learns projection vectors instead of the random projections, learning to hash has been attracting a large amount of research interest in computer vision and machine learning. With the development of deep learning [75], getting hashing code through deep learning gets more and more attention for two reasons. The first reason is that the powerful representation capabilities of deep learning can learn very complex hash functions. The second reason is that deep learning can achieve end-to-end hashing codes, which is very useful in many applications. In this survey, we mainly focus on deep supervised hashing methods and some other topics are also included.

The design of the deep supervised hashing method mainly includes two parts, namely the design of the network structure and the design of the loss function. For small dataset like MINST [76] and CIFAR-10 [71], shallow architecture such as AlexNet [73] and CNN-F [19] are widely used. While for complex dataset like NUS-WIDE [25] and COCO [86], deeper architecture such as VGG [107] and ResNet50 [50] are needed. The intuition of the loss function design is to maintain similarity, such as minimizing the gap between the similarity in the original space and the similarity in the hash space. The similarity in the original space is usually obtained by using semantic label information or the distance relationship in the original space, which is widely studied in different deep hashing methods. Hence we mainly focus on the similarity preserving manners latter.

Following to [116], we also categorizes the deep hashing algorithms according to the similarity preserving manners into: pairwise similarity preserving, multiwise similarity preserving, implicit similarity preserving, quantization and classification oriented. For each manner, We comprehensively analyzed how the related article designs the loss function, takes advantages of semantic labels and what additional tricks are used. In addition, we also introduce some other topics such as deep unsupervised hashing and multi-modal deep hashing methods. Meanwhile, we also present some commonly used public datasets and the scheme to measure the perfor-

mance of deep hashing algorithms. At last, a comparison of some key algorithms was given.

In comparison to other surveys on hash [115, 114, 116, 15], this survey mainly focuses on deep hashing methods and how they design loss functions. As far as we know, this is the most comprehensive survey about deep hashing, which is helpful for readers to understand the mechanisms and trends of deep hashing.

2 Background

2.1 Nearest neighbor search

Given a d-dimensional Euclidean space $\mathbb{R}^d$, the nearest neighbor search problem is to find the element $NN(x)$ in a finite set $\mathcal{Y} \subset \mathcal{R}^d$ with n points such that

$$NN(x) = \text{argmin}_{y \in \mathcal{Y}} \rho(x, y), \quad (1)$$

where $x \in \mathbb{R}^D$ is called a query. The distance ρ can be by Euclidean distance, general ℓ_p distance, cosine similarity and so on. Many exact nearest neighbor search method such as KD-tree [35] were developed by researcher, which works quite well when d is small. However, Nearest neighbor search is inherently expensive due to the curse of dimensionality [2] [3]. Although KD-tree can be extended to high-dimensional situations, the effect is not very good, even slower than brute force search.

To solve this problem, a series of algorithms for approximate nearest neighbors have been proposed [28, 40, 97, 62]. The principle of these methods is to find the nearest point with a high probability, rather than to find the nearest point accurately. These ANN algorithms are mainly divided into three categories: hashing-based [28, 1, 93], product quantization based [62, 37, 68, 132] and graph based [46, 94, 95]. These algorithms have greatly improved the efficiency of searching while ensuring a relatively high accuracy rate, so they are widely used in the industry. Compared to the other two types of methods, hash-based algorithms are the longest studied and the most studied by people at the same time. Because it has great potential in improving computing efficiency and reducing memory cost.

2.2 Search with hashing

The Purpose of the hash algorithm is to map the features of the original space into a Hamming space, which results in a short compact hashing code consists of 0 and 1. Because the computer's calculations and storage are implemented in binary, hash coding is very efficient in storage and calculation. There are two main types

of hash-based search algorithms: hash table lookup and hash code ranking.

The main idea of hash table lookup for accelerating the search is reducing the number of the distance computations. The data structure, called hash table (a form of inverted index), is composed of buckets with each bucket indexed by a hash code. Each point is assigned to a hash bucket which shares the same hash code. Therefore, the strategy for learning hash encoding for this type of algorithm is to make the relatively close points in the original space have a higher probability of having the same hash encoding. When a query comes, we can find the corresponding hash bucket according to the hash code of the query, so as to find the corresponding candidate set. After this step, we usually re-rank the points in the candidate set to get the final search target. However, the recall of selecting a single hash bucket as a candidate set will be relatively low. To overcome this problem, two methods are usually adopted. The first method is to select some buckets that are close to the target bucket at the same time. The second method is to independently create multiple different hash tables according to different hash codes. Then we can select the corresponding target bucket from each hash table.

Hash code ranking is a relatively easier way than hash table lookup. When a query comes, we simply compute the Hamming distance between query and each points in the searching dataset. Then select the points with relative smaller Hamming distance as the candidates for nearest neighbor search. After that, a re-ranking process by the original features is usually followed to obtain the final nearest neighbor. Differ to hash table lookup methods, Hash code ranking methods prefer hash codes that preserve the similarity/distance of the original space.

2.3 Deep neural network

Deep Convolutional Neural Networks (CNNs), a special type of Neural Networks, which have shown significant performance improvement in several Image Processing and Computer Vision competitions, such as ImageNet [73]. The powerful and effective learning ability of deep CNN is mainly derived from the utilization of multiple feature extraction stages that can automatically learn feature representations from the origin images. In 2012, A. Krizhevsky et al. [73] drew attention to the public by AlexNet network which achieved a top-5 error rate of 15.3% outperforming the previous best traditional model in ImageNet dataset. Since the 2012 milestone, many researcher have tried to go deeper in the sequences of convolution layers to achieve

better performance. In 2014, K. Simonyan & A. Zisserman [108] introduced the 16-layers VGG model that chains multiple convolution layers to win this competition. The same year, M. Lin et al. [85] have developed the concept of inception modules which is further exploited by C. Szegedy et al. [112] who proposed a deeper network called GoogLeNet with 22 layers. The main common trend to design convolutional neural network models is the increasing network depth. However, as the depth increased, networks involved an increasing error rate due to the difficulties to optimize an extremely deep models. K. He et al. [51] proposed ResNet network by residual learning to further deepen network to 101 layers. To avoid the tedious network architectures design, Google Brain researchers (B. Zoph and Q.V. Le, 2017) [138] have proposed a new concept called Neural Architecture Search(NAS) which searches state-of-the-art networks automatically. After that, various NAS networks have been released to the community, such as PNAS [88], ENAS [100], EfficientNet [113] and so on. And these classic network architectures further become the backbone networks in other tasks, such as image retrieval [116] and object detection [47].

3 Deep Supervised Hashing

3.1 Learning to Hash

Given an input item $\mathbf{x}$, learning to hash is the task of learning a hash function f, which maps $\mathbf{x}$ to a compact hash code $\mathbf{b}$ for the convenience of the nearest neighbor search. The hash code obtained by a good hash function should preserve the distance order in the original space as much as possible, i.e. those items that are close to query by hash code should also be close to query in the original space. Many traditional hash functions including linear projection, kernels, spherical function, a non-parametric function[116, 110, 48] were proposed by researchers to learn compact hash codes, and achieved significant progress. However, these simple hash function do not work well for large dataset. For the strong representation ability of deep learning, more and more researchers pay attention to deep supervised hashing and develop a lot of new methods. And these methods achieve higher performance than traditional methods.

Deep supervised hashing uses deep neural networks as hash functions, which can generate hash code end-to-end. A good deep supervised hashing model usually needs to consider four issues: what deep neural network architecture is adopted, how to take advantage of the similarity and semantic(class) information, how to train the neural network with the discretization prob-

lem and what other skills can be used to improve the performance.

3.2 Network Architecture

Traditional hashing methods usually utilize linear projection and kernels, which shows poor representation ability. After AlexNet and VGGNet[73,107] were proposed, deep learning shows its superiority in computer vision, especially for classification problems. And more and more experiments proved that the deeper the network, the better the performance. As a result, ResNet [50] takes advantage of residual learning, which can train very deep networks, achieved significantly better results. After that, ResNet and its variants become basic architectures in deep learning[50,55,121]. The latest researches often utilize the popular architectures with pre-trained weights in the large datasets such as ImageNet, following the idea of transfer learning. Most of researches utilize the shallower architectures such as AlexNet, CNN-F and designed stacked convolutional neural networks for simple datasets, e.g. MINST, CIFAR-10. Deeper architectures such as VGGNet and ResNet50 are often utilized for complex datasets such as NUS-WIDE and COCO. For deep supervised hashing methods, the last layer (i.e. classification layer) is often replaced by a hash layer, which has a dense connection with the feature layer. And the hash code can be obtained by the output of hash layer with a sign activation.

The network architecture is one of the most important factors for deep supervised hashing, and it affects both the accuracy of the search and the time cost of inference. If the architecture is degenerated into MLP or linear projections, the deep supervised hashing become the traditional hashing methods. Although the deeper the network architecture, the greater the search accuracy, but it also increases the time cost. We think that the architecture need to be considered combined with the complexity of datasets. However, all the deep hashing methods can change their architectures without any the other change. Therefore, we do not use the network architecture to categorize the deep supervised hashing algorithms.

3.3 Similarity

In the rest, we always define $\mathbf{x}_i$ as the input, $\mathbf{h}_i$ as the output of the network, $\mathbf{b}_i$ as the obtained binary codes, for each sample i. We denote the distance between pair of items $(\mathbf{x}_i, \mathbf{x}_j)$ in the input space and hash coding space as s_{ij}^o and s_{ij}^h, respectively. In the input space the similarity is the ground truth, which mainly includes items distance d_{ij}^o and semantic similarity. The former is the distance of features, e.g. Euclidean distance $\|\mathbf{x}_i - \mathbf{x}_j\|_2$ and the similarity can be defined by Gaussian function or Characteristic function, i.e. $\exp\left(-\frac{\left(d_{ij}^o\right)^2}{2\sigma^2}\right)$, $I_{d_{ij}^o < \tau}$, in which τ is a given threshold. The cosine similarity is also popular. The similarity is usually binary in deep supervised hashing, where the value is 1 if the two items have common semantic label and 0 vice visa.

In the hash coding space, the distance d_{ij}^h is Hamming distance naturally, which is defined as the number of bits where the values are different and is formulated as:

$$d_{ij}^h = \sum_{m=1}^{M} \delta\left[b_{im} \neq b_{jm}\right],$$

If the code is valued by 1 and 0, we have

$$d_{ij}^h = \|\mathbf{b}_i - \mathbf{b}_j\|_1,$$

and it varies from 0 to M. As a result, the similarity based on the Hamming distance is defined as $s_{ij}^h = M - d_{ij}^h$. If the code is valued by 1 and -1, we have

$$d_{ij}^h = \frac{1}{2}(M - \mathbf{b}_i^T \mathbf{b}_j),$$

The similarity is defined by the inner product, i.e. $s_{ij}^h = \mathbf{b}_i^\top \mathbf{b}_j$. These measures can also be extended to the weighted cases. i.e.

$$d_{ij}^h = \sum_{m=1}^{M} \lambda_m \delta\left[b_{im} \neq b_{jm}\right],$$

in which each bit has a weight λ_m, and if the codes are valued by 1 and -1, we have

$$s_{ij}^h = \mathbf{b}_i^\top \varLambda \mathbf{b}_{j'},$$

where $\mathbf{\Lambda} = Diag\left(\lambda_1, \lambda_2, \ldots, \lambda_M\right)$ is a diagonal matrix and each diagonal entry is the weight of the corresponding hash bit.

3.4 Loss Function

A good loss function is one of the factors for the success of deep supervised hashing. The basic rule of designing the loss function is to preserve the similarity order, i.e. minimizing the gap between the similarity in the original space and the similarity in the hash space. As a result, almost all the loss functions contain the

terms of similarity information. For example, the typical loss function is pair-wise loss, making similar images have similar hash codes (small hamming distance) and dissimilar images have dissimilar hash codes (large hamming distance). Besides, the multi-wise similarity preserving loss term, especially in triplet forms, making the orders among multiple items computed from the original and new spaces as consistent as possible, is also widely utilized. There are also several implicit and other variants similarity preserving loss terms.

Besides similarity information, the semantic(label) information is also included for the design of loss function. There are three popular ways to take advantage of label information that are summarized below. The first way is regression on hash codes with label. The label is encoded into one-hot format matrix and regression loss, i.e.$||Y - WH||_F$ are added into loss function. The second way is adding a classification layer after the hash layer, and classification loss(cross-entropy loss)are added to the loss function. The last one is utilizing Lab-Net, which was first proposed in [79]. The principle of LabNet is explored to capture abundant semantic correlation between sample pairs.

The quantization loss term is also common in deep supervised hashing, especially in quantization-based hashing methods. There are also several skills can be added in loss function, e.g. bit balancing, regularization, and so on, which is also important for improving the performance.

3.5 Optimization

The challenges for optimizing the neural network parameters is the vanishing gradient problem form sign function which is used to obtain binary hash codes. Specifically, the gradient of sign function is zero for all nonzero input, and that is fatal to the neural network which uses gradient descent for training.

Almost all the works adopt that continuous relaxation by replacing sign function with tanh or sigmoid, and later in test phase apply sign function to obtain final binary codes. The first typical way is quantization function by adding a penalty term in loss function, which is often formulated as $|||\mathbf{h}_i| - \mathbf{1}||_1$, or $-||\mathbf{h}_i||$ with tanh activation. This penalty term helps the neural network to obtain $sgn(z) \approx z$. It's noticed that this loss can be considered as a novel prior for each hash code h based on a symmetric variant of some distribution, e.g. bimodal Laplacian and Cauchy distribution. From this view we can get some variants, e.g. pairwise quantization[137] and Cauchy quantization loss[12]. If the loss function is non-smooth function whose derivative is difficult to compute, a smooth surrogate can be

adopted, e.g. $|x| \approx \log(\cosh x)$, which helps get the smooth loss function[137]. The second way is an alternative scheme, which decompose the optimization into several sub-problems, which can be iteratively solved by using the alternating minimization method. In this alternative process, back propagation can only works in one sub-problem and the other sub-problems can be solved by other optimization methods. For example, DSDH[80] utilizes the discrete cyclic coordinate descend algorithm. This methods can keep the discrete constraint during the whole optimization process while it can not lead to end-to-end training, which have limited application for solving the unknown sub-problems. The third method is named continuation which utilizes a smooth activation function $y = tanh(\beta x)$ to approximate the discrete sign function by increasing β [16]. There are some other ways to solve this problem by changing the calculation and the propagation of gradients, e.g. Greedy Hash[111] and Gradient Attention Network[56], which improved the effectiveness and accuracy of deep supervised hashing.

3.6 Categorization

Our survey categorizes the existing algorithms to the following five classes base on the similarity preserve manners: the pairwise similarity preserving class, the multi-wise similarity preserving class, the implicit similarity preserving class, the quantization class and classification oriented class. We separate the quantization class from the pairwise similarity preserving class similar to [116]. For each class, we will discuss the corresponding deep hashing methods in detail one by one. The summary of these algorithms is shown in Table 1.

The main reason we choose the similarity preserving manner to do the categorization is that similarity preservation is the essential goal of hashing, which is almost essential in loss function in deep supervised hashing. Other factors such as architecture, label information, optimization as well as other skills is also significant for the performance.

4 PAIRWISE SIMILARITY PRESERVING

The algorithms aligning the distances or similarities of a pair of items computed from the input space and the Hamming coding space are roughly divided into the following groups:

- Product loss minimization: The loss is in the product form of the similarity information between the input space and hash coding space. The similarity information includes distance and similarity. For

Table 1 A Summary of Representative Deep Supervised Hashing Algorithms with Respect to the Manner to Utilize Similarity Information, Handle the Label Information as well as the sgn Function, Other Tricks. Drop = drop the sgn operator in the neural network and regard the hash code as a discrete approximation of the hash value, Two step = two-step optimization

Approach	Similarity Information	Label	sgn	Other skills
SDH[34]	$s_{ij}^o d_{ij}^h$	-	Quantization Loss	Bit Balance+Orthogonality
DSH[89]	$s_{ij}^o d_{ij}^h$+Margin	-	Quantization Loss	-
PCDH[23]	$s_{ij}^o d_{ij}^h$+Margin	Adding Classification Layer	Drop	Pairwise correlation loss
WMRSH[78]	$s_{ij}^o d_{ij}^h$+Margin	Adding Classification Layer	Quantization Loss	Bit and table weight
SHBDNN[32]	$\left(s_{ij}^o - s_{ij}^h\right)^2$	-	Quantization Loss+Alternation	Bit Balance+Independence
DDSH[64]	$\left(s_{ij}^o - s_{ij}^h\right)^2$	-	Alternation	Splitting the training set
CNNH[120]	$\left(s_{ij}^o - s_{ij}^h\right)^2$	Part of hash code	-	Two-step
AsDSH[66]	$\left(s_{ij}^o - s_{ij}^h\right)^2$	-	Quantization Loss +Alternation	Asymmetry
DIH[119]	$\left(s_{ij}^o - s_{ij}^h\right)^2$	-	Quantization Loss+Alternation	Incremental part+Bit balance
HBMP[8]	$\left(s_{ij}^o - s_{ij}^h\right)^2$	-	Drop	Bit Weight+Two-step
DOH[67]	$\left(s_{ij}^o - s_{ij}^h\right)^2$	-	Ranking	FCN
DPSH[81]	$\log(1 + \exp s_{ij}^h) - s_{ij}^o s_{ij}^h$	-	Quantization Loss	-
DHN[137]	$\log(1 + \exp s_{ij}^h) - s_{ij}^o s_{ij}^h$	-	Quantization Loss+Smooth	-
HashNet[16]	$w_{ij} \log(1 + \exp \alpha s_{ij}^h) - \alpha s_{ij}^o s_{ij}^h$	-	tanh+Continuation	-
DSDH[80]	$\log(1 + \exp s_{ij}^h) - s_{ij}^o s_{ij}^h$	Linear Regression+L2	Quantization Loss +Alternation	-
DAPH[104]	$\log(1 + \exp s_{ij}^h) - s_{ij}^o s_{ij}^h$	-	Quantization Loss+Alternation	Bit Balance+Independence
DAgH[125]	$\log(1 + \exp s_{ij}^h) - s_{ij}^o s_{ij}^h + \left(s_{ij}^o - s_{ij}^h\right)^2$	-	-	Two-step
DCH[12]	$w_{ij}\left(s_{ij} \log \frac{s_{ij}^h}{\gamma} + \log\left(1 + \frac{\gamma}{s_{ij}^h}\right)\right)$	-	Cauchy Quantization Loss	-
DJSEH[79]	$\log(1 + \exp s_{ij}^h) - s_{ij}^o s_{ij}^h$	LabNet+Linear Regression	Quantization Loss	Two-step+Asymmetry
ADSQ[127]	$\left(s_{ij}^o - s_{ij}^h\right)^2 + \log(1 + \exp s_{ij}^h) - s_{ij}^o s_{ij}^h$	LabNet	Quantization Loss+Alternation	Bit balance loss+Two-step
MMHH[69]	$w_{ij} s_{ij}^o \log\left(1 + \max\left(0, d_{i,j}^h - H\right)\right)$	-	Quantization Loss	Semi-Batch Optimization
DAGH[22]	$\log(1 + \exp s_{ij}^h) - s_{ij}^o s_{ij}^h$	Linear Regression	Drop+Alternation	Regression with Anchor Graph
HashGAN[11]	$w_{ij} \log(1 + \exp \alpha s_{ij}^h) - \alpha s_{ij}^o s_{ij}^h$	-	Cosine Quantization Loss	GAN
DPH[17]	$w_{ij} \log(1 + \exp \alpha s_{ij}^h) - \alpha s_{ij}^o s_{ij}^h$	-	Priority Quantization Loss	Priority Cross-Entropy Loss
DFH[82]	$s_{ij}^o \log\left(1 + \exp d_{ij}^h\right)$+Margin	-	Quantization Loss+Alternation	Quantized Center Loss
DRSCH[131]	$s_{ij}^o d_{ij}^h$+Margin+Triplet Loss	-	Drop	Bit Weight
DNNH[74]	Triplet ranking loss+Margin	-	Piece-wise threshold function	-
DSRH[135]	Surrogate loss	-	Quantization Loss	Bit Balance
DTSH[117]	Triplet Loglikelihood Loss	-	Quantization Loss	-
DSHGAN[101]	Triplet loss+Margin	Adding Classification Layer	Drop	GAN
AnDSH[136]	Angular Hashing Loss	Adding Angular-softmax	Drop	Bit balance
HMI[7]	Mutual Information	-	Drop	Minibatch Backpropagation
HTALR[49]	-	-	-	Tie-Awareness
MLRDH[92]	-	Multi-linear Regression	Alternation	Hash Boosting
CSH[130]	-	Central Similarity Loss	Quantization Loss	-
HCBDH[21]	-	Adding Classification Layer	-	Hadamad loss
DBH[84]	-	Adding Classification Layer	-	Transform Learning
SSDpH[126]	-	Adding Classification Layer	Quantization Loss	Bit Balance
VDSH[134]	-	Linear Regression	Drop+Alternation	-
SUBIC[60]	-	Adding Classification Layer	Drop+Entropy Regularization	Bit Balance
DVsQ[13]	Triplet Loss+Margin	-	Inner-Product Quantization Loss	Learning label embeddings
DPQ[70]	-	Adding Classification Layer	-	Joint Central Loss
DSQ[33]	-	Adding Classification Layer	Quantization+Alternation	Joint Central Loss
SPDAQ[20]	$\left(s_{ij}^o - s_{ij}^h\right)^2$	Adding Classification Layer	Drop + Alternation	Asymmetry
DQN[14]	$\left(s_{ij}^o - s_{ij}^h\right)^2$	-	Product Quantization Loss	Asymmetric Quantizer Distance
DTQ[87]	Triplet ranking loss+Margin	-	Weak-Orthogonal Quantization	Group Hard
EbDSH[45]	-	-	-	Ensemble Learning

example, similarity-distance product minimization, i.e. $\min \sum_{(i,j) \in \mathcal{E}} s_{ij}^o d_{ij}^h$, which expects the distance in the coding space to be smaller if the similarity in the original space is larger. In the formulation, $\mathcal{E}$ is a set of pair items that are considered. It is evident to see that there are three other forms of loss function, which are similar to the given one.

– Difference loss minimization: The loss is in difference form minimize the difference between the similarities or distances , i.e., $\min \sum_{(i,j) \in \mathcal{E}} \left(d_{ij}^o - d_{ij}^h\right)^2$ or $\min \sum_{(i,j) \in \mathcal{E}} \left(s_{ij}^o - s_{ij}^h\right)^2$.

– Likelihood loss minimization: The kind of loss is derived from probabilistic model. Given similarity matrix $S = s_{ij}^o$ and hash codes H, the total likelihood function of hash codes can be represented as

$$p(\mathbf{H}|\mathbf{S}) \propto p(\mathbf{S}|\mathbf{H})p(\mathbf{H}) = \prod_{(i,j) \in \mathcal{E}} p\left(s_{ij}^o|s_{ij}^h\right) p(s_{ij}^h),$$
(2)

where $p(\mathbf{S}|\mathbf{H})$ denotes the likelihood and $p(\mathbf{B})$ is the prior distribution. $p\left(s_{ij}^o|s_{ij}^h\right)$ is the conditional probability of s_{ij}^o given their hash codes. In formulation,

$$p\left(s_{ij}^o|s_{ij}^h\right) = \begin{cases} \sigma\left(s_{ij}^h\right), & s_{ij}^o = 1 \\ 1 - \sigma\left(s_{ij}^h\right), & s_{ij}^o = 0 \end{cases}$$
(3)

in which $\sigma(x) = 1/(1 + e^x)$. From the formulation, the probabilistic model expects the similarity in the

coding space to be larger if the similarity in the original space is larger. The loss function is the negative log-likelihood, i.e.

$$-\log p(\mathbf{S}|\mathbf{H}) = \sum_{(i,j)\in\mathcal{E}} \log(1 + e^{s_{ij}^h}) - s_{ij}^o s_{ij}^h. \quad (4)$$

Next, we will review these three types of deep hashing methods in turn. Please note that the above formulations can have a lot of variants for different consideration.

4.1 Product loss minimization

Deep Supervised Hashing[89] The network of DSH consists of three convolutional-pooling layers and two fully connected layers. The origin pairwise loss function is defined as:

$$\mathcal{L}(s_{ij}^o, d_{ij}^h) = \frac{1}{2}s_{ij}^o d_{ij}^h + \frac{1}{2}(1 - s_{ij}^o)max(m - d_{ij}^h, 0) \quad (5)$$
$$s.t. \mathbf{h}_i, \mathbf{h}_j \in \{-1, 1\}^k,$$

where $m > 0$ is a margin threshold parameter. d_{ij}^h denotes the Hamming distance between two binary vectors and $s_{ij}^o = 1$ if the h_i, h_j have the same semantic label and 0 vice visa. The loss function is in the form of distance-similarity product minimization which punishes similar images mapped to similar hash codes and dissimilar images mapped to different hash codes when their Hamming distance falls below the margin threshold m. It is noticed that when d_{ij}^h is larger than m, the loss does not punish this pair. This idea is similar to hinge loss function.

As we discuss before, DSH relaxes the binary constraints and imposes a regularizer on the real-valued network outputs to approximate the binary codes, i.e. $h \approx sgn(h)$. The pairwise loss is rewritten as:

$$\frac{1}{2}s_{ij}^o||\mathbf{h}_i - \mathbf{h}_j||_2^2 + \frac{1}{2}(1 - s_{ij}^o)max(m - ||\mathbf{h}_i - \mathbf{h}_j||_2^2, 0)$$
$$+ \alpha \sum_{k=i,j} |||\mathbf{h}_k| - \mathbf{1}||_1, \quad (6)$$

where $\mathbf{1}$ is the vector of all ones, $||\cdot||$ is the ℓ_1-norm of vector, $|\cdot|$ is the element-wise absolute value operation and α is a weighting parameter that controls the strength of the regularizer. DSH doesn't utilize saturating non-linearities because they think that it may slow down the training process. With the above loss function, the neural network is able to be trained with

end-to-end BP algorithm. For the test samples, the binary codes can be obtained by sign function.

DSH is a simple and straight-forward deep supervised hashing method in the early period, and its idea is originated from Spectral Hashing[118] but with deep learning framework. It is easy to understand but its performance is limited.

Pairwise Correlation Discrete Hashing[23] PCDH utilizes four fully connected layers after convolutional-pooling layer, named deep feature layer, hash-like layer, discrete hash layer as well as classification layer respectively. The third layer can directly generate discrete hash code. Differ to DSH, PCDH leverages ℓ_2 norm of deep features and hash-like codes. Besides, classification loss is also added to the loss function:

$$\mathcal{L} = \mathcal{L}_s + \alpha\mathcal{L}_p + \beta\mathcal{L}_l$$
$$= \sum_{(i,j)\in\mathcal{E}} (\frac{1}{2}(1 - s_{ij}^o)\max(m - ||\mathbf{b}_i - \mathbf{b}_j||_2^2, 0)^2$$
$$+ \frac{1}{2}s_{ij}||\mathbf{b}_i - \mathbf{b}_j||_2^2)$$
$$+ \alpha \sum_{(i,j)\in\mathcal{E}} (\frac{1}{2}(1 - s_{ij}^o)\max\left(m - ||\mathbf{w}_i - \mathbf{w}_j||_2^2, 0\right)^2$$
$$+ \frac{1}{2}s_{ij}^o||\mathbf{w}_i - \mathbf{w}_j||_2^2)$$
$$+ \beta(\sum_{i=1}^N \phi(w_{ic}^T\mathbf{b}_i, \mathbf{y}_i) + \sum_{j=1}^N \phi(w_{jc}^T\mathbf{b}_j, \mathbf{y}_j))$$
$$(7)$$

where $\mathbf{w}_i, \mathbf{b}_i$ and $\mathbf{h}_i$ is the output of the first three fully connected layer and the last term is the classification cross-entropy loss. It's noticed that the second term is called pairwise correlation loss. PCDH also guides the similarity of deep features, which avoids overfitting compared with DSH. And the classification loss provides the semantic supervision, which helps the model achieving competitive performance. Besides, PCDH proposes a pairwise construction module named Pairwise Hard, which samples positive pairs with maximum distance between deep features and negative pairs with distance smaller than the threshold randomly. It is evident that Pairwise Hard chooses the pairs with large loss for effect hash codes learning.

Supervised Deep Hashing[34] utilizes the fully-connected neural network for deep hashing and has a similar loss function except for a term that enforces a relaxed orthogonality constraint on all projection matrices(i.e. weight matrices in neural network) for the property of fully-connected layers. Bit balance regularization is also included which will be introduced below.

4.2 Difference loss minimization

Supervised Hashing with Binary Deep Neural Network[32] The architecture of SH-BDNN is stacked by fully connected layer, in which $\mathbf{W}_i$ is the weight of the i-th layer. SH-BDNN does not only considers the similarity information, but also considers independence of different hash bits, i.e. each bit has a 50% chance of being 1 or -1. Given the hash code matrix $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_N]^T$, the two conditions are formulated as,

$$\mathbf{B}^T \mathbf{1} = 0, \frac{1}{N} \mathbf{B}^T \mathbf{B} = \mathbf{I}$$

,where $\mathbf{1}$ is an N-dimension vector whose elements are all one, and $\mathbf{I}$ is an identity matrix of size N by N.

The loss function is

$$
\begin{aligned}
\mathcal{L} = &\frac{1}{2N} \left\| \frac{1}{l} \left(\mathbf{H} \right) \mathbf{H}^T - \mathbf{S} \right\|^2 + \frac{\lambda_1}{2} \sum_{l=1}^{n-1} \left\| \mathbf{W}^{(l)} \right\|^2 \\
&+ \frac{\lambda_2}{2N} \left\| \mathbf{H} - \mathbf{B} \right\|^2 + \frac{\lambda_3}{2} \left\| \frac{1}{N} \mathbf{H}^T \left(\mathbf{H} \right) - \mathbf{I} \right\|^2 \\
&+ \frac{\lambda_4}{2m} \left\| \mathbf{H}^T \mathbf{1} \right\|^2 \\
&\text{s.t. } \mathbf{B} \in \{-1, 1\}^{N \times l},
\end{aligned}
\tag{8}
$$

where l is the length of hash code and N is the sample size. $\mathbf{H}$ is the output of network and $\mathbf{B}$ is the binary hash codes. $\mathbf{S}$ is the pairwise similarity matrix valued 1 or -1. The First term is similarity difference loss minimization, the second term is the ℓ_2 regularization, the third term is the quantization loss, the last two terms are to punish the bit's dependency and imbalance respectively. It's noticed that the $\mathbf{B}$ is not the sign of $\mathbf{H}$. As a result, the loss function is optimized by updating the network parameter and $\mathbf{B}$ alternatively. SH-BDNN has a well-designed loss function which follows Kernel-based Supervised Hashing[90]. However, the architecture doesn't include the popular convolutional neural network and ti is not an end-to-end model. As a result, the performance of this model is limited in complex datasets.

Convolutional Neural Network Hashing[120] CNNH is the first deep supervised hashing method to my knowledge. CNNH adopted a two-step strategy. In step one, it utilizes coordinate descent algorithm to optimize loss function

$$\mathcal{L} = \left\| \frac{1}{l} \left(\mathbf{H} \right) \mathbf{H}^T - \mathbf{S} \right\|^2 \tag{9}$$

in order to learn approximate hash codes. In step two, CNNH utilizes obtained hash codes to train the convolutional neural network with l output units. Besides, if class labels is available, the fully connected output layer is added K output units which corresponds the K class labels of images and the classification loss is added to loss function. Although the way CNNH uses labels looks very clumsy, this two-step strategy is still popular in deep supervised hashing and inspired many other state-of-the-art methods.

Hashing with Binary Matrix Pursuit[8] HBMP also takes advantage of the two-step strategy introduced above. Differ to CNNH, HBMP utilizes weighted Hamming distance and adopts a different traditional hashing algorithm called binary code inference to get hash codes. In the first step, the loss function is

$$\mathcal{L} = \frac{1}{4} \sum_{i,j} [\mathbf{b}_i^T \Lambda \mathbf{b}_j - s_{ij}^o]^2, \tag{10}$$

where $\mathbf{\Lambda}$ is a diagonal matrix $Diag(\lambda_1, \ldots, \lambda_l)$. It is noticed that the similarity matrix $\mathcal{S}^h$ with $\mathcal{S}_{ij}^h = \mathbf{b}_i^T \Lambda \mathbf{b}_j$ can be approximated by step-wise algorithm. HBMP also trains a convolutional neural network by the obtained hash codes with point-wise hinge loss and shows that deep neural network helps to simplify optimization problems and get robust hash codes.

Deep Discrete Supervised Hashing[64] DDSH uses a column-sampling method to split the whole training set into two parts, $\mathbf{X}^\Omega$ and $\mathbf{X}^\Gamma$, where Ω and Γ are the indexes. The loss function is designed by an asymmetric form:

$$\mathcal{L} = \sum_{i \in \Omega, j \in \Gamma} \mathcal{L}(\mathbf{b}_i, \mathbf{h}_j, S_{ij}^o) + \sum_{i,j \in \Omega} \mathcal{L}(\mathbf{b}_i, \mathbf{b}_j, S_{ij}^o), \tag{11}$$

where $\mathbf{b}_i$ and $\mathbf{h}_i$ are the binary codes and output of network, respectively. And $\mathbf{b}_i$ and $\mathbf{h}_i$ are updated alternatively just like [32]. It is significant that DDSH provides an asymmetric perspective of learning to hash and utilizes pairwise similarity information to directly guide both discrete coding generating and deep feature learning.

Asymmetric Deep Supervised Hashing[66] ADSH considers the database points and query points in an asymmetric way, which can help to train the model more effectively. And it can also help to achieve better performance. ADSH contains two important components: feature learning part and loss function part. The first one is to utilize a deep neural network to learn binary hash codes for query points. And the second one is used to directly learn hash codes for database points by optimizing the loss function with supervised information. The loss function is formulated as:

$$\mathcal{L} = \sum_{i \in \Omega, j \in \Gamma} [tanh(F(\mathbf{x}_i; \Theta)^T \mathbf{b}_j - l s_{ij}^o]^2, \tag{12}$$
$$\text{s.t.} \mathbf{b}_j \in \{-1, 1\}^l,$$

where Ω is the index of query points, Γ is the index of database points and $F(\cdot; \Theta)$ is the neural network with parameter Θ. Θ and $\mathbf{b}_j$ are updated alternatively during the optimization process. If only the database points are available, we let $\Omega \subset \Gamma$ and add a quantization loss $\gamma \sum_{i \in \Omega} [\mathbf{h}_i - tanh(F(\mathbf{x}_i; \Theta))]^2$. This asymmetric strategy by combining deep hashing and traditional hashing can help to achieve better performance.

Deep Incremental Hashing Network[119] DIHN tries to learn hash codes in an incremental manner. Similar to ADSH[66], the dataset was divided into two parts: original and incremental databases respectively. When a new image comes from incremental database, its hash code is learned depend on the hash codes of original database. The optimization process still uses the strategy of alternately updating parameters.

Deep Ordinal Hashing[67] DOH learns ordinal hash codes by taking advantage of both local and global features. Specifically, two subnetwork learn the local spatial information from Fully Convolutional Network with spatial attention module and the global semantic information from the Convolutional Neural Network, respectively. Afterwards, the two outputs are combined by dot product to produce R ordinal outputs $\mathbf{h_i}$. For each segment $\mathbf{h_i}$, the corresponding hash code can be obtained by

$$b_i^r = argmax_\theta \theta^T \mathbf{h}_i, \tag{13}$$
$$\text{s.t.} \theta \in \{0, 1\}^K, \|\theta\|_1 = 1.$$

And the full hash code can be obtained by concatenating all b_i^rs. DOH adopts an end-to-end ranking-to-hashing framework, which avoiding using undifferentiable sign functions. What's more, it uses a relatively complex network that is enable to handle large datasets with high performance.

4.3 Likelihood loss minimization

Deep Pairwise Supervised Hashing[81] DPSH uses CNN-F[19] as the basic network framework and the standard form of likelihood loss based on similarity information. Besides similarity information, quantization loss is also added to the loss function, i.e.

$$\mathcal{L} = - \sum_{(i,j) \in \mathcal{E}} \left(s_{ij}^o s_{ij}^h - \log \left(1 + e^{s_{ij}^h} \right) \right)$$
$$+ \eta \sum_{i=1}^n ||\mathbf{h}_i - sgn(\mathbf{h}_i)||_2^2, \tag{14}$$

where $s_{ij}^h = \frac{1}{2}\mathbf{h}_i^T \mathbf{h}_j$ and $\mathbf{h}_i$ is the direct output of the network. Although triplet loss was popular at that time, DPSH adopted the pairwise form simultaneously learned deep features and hash codes, which improved both accuracy and efficiency. This likelihood function-based loss function can easily introduce different Bayesian priors, making it very flexible in application and achieving better performance than the other two kinds of loss.

Deep Hashing Network[137] has similarity loss function with DPSH. Differ to DPSH, DHN sees quantization loss as Bayesian prior and propose a bimodal Laplacian prior for the output $\mathbf{h}_i$, i.e.

$$p(\mathbf{h}_i) = \frac{1}{2\epsilon} \exp \left(-\frac{\|\,|\mathbf{h}_i| - \mathbf{1}\|_1}{\epsilon} \right), \tag{15}$$

and this negative log likelihood (i.e. quantization loss)is

$$\mathcal{Q} = \sum_{i=1}^n ||\,|\mathbf{h}_i - \mathbf{1}||_1, \tag{16}$$

and it can be smoothed by a smooth surrogate[58] into

$$\mathcal{Q} = \sum_{i=1}^n \sum_{k=1}^l log(cosh(|h_{ik}| - 1)), \tag{17}$$

where h_{ik} is the k-th element of $\mathbf{h}_i$. We notice that the DHN replaced ℓ_2 norm(ITQ quantization error[41]) by ℓ_1 norm. And they also show that the ℓ_1 norm is an upper bound of the ℓ_2 norm, and the ℓ_1 norm encourages sparsity and is easier to optimize.

HashNet[16] As a variant of DHN, HashNet considered the imbalance training problem that the positive pairs are much more than the negative pairs. So it adopted Weighted Maximum LikeLihood(WML) loss with different weights for each pair. The weight is formulated as

$$w_{ij} = c_{ij} \cdot \begin{cases} |\mathcal{S}| / |\mathcal{S}_1|, & s_{ij}^o = 1 \\ |\mathcal{S}| / |\mathcal{S}_0|, & s_{ij}^o = 0 \end{cases} \tag{18}$$

where $\mathcal{S}_1 = \left\{ (i, j) \in \mathcal{E} : s_{ij}^o = 1 \right\}$ is the set of similar pairs and $\mathcal{S}_0 = \mathcal{E}/\mathcal{S}_1$ is the set of dissimilar pairs. $c_{ij} = \frac{\mathbf{y}_i \cap \mathbf{y}_j}{\mathbf{y}_i \cup \mathbf{y}_j}$ for multi-label datasets and equals 1 for single-label datasets. Besides, the sigmoid function in condition probability is substituted by $1/1 + e^{-\alpha x}$ called adaptive sigmoid function which equals adding a hyperparameter into the hash coding similarity $s_{ij}^h = \alpha \mathbf{b}_i^T \mathbf{b}_j$. Different from other methods, HashNet continuously approximates sign function through tanh functions

$$\lim_{\beta \to \infty} \tanh(\beta z) = \text{sgn}(z).$$

The network output activation function is $\tanh(\beta_t \cdot)$ by evolving with $\beta_t \to \infty$ step-wise and the network will

converge to HashNet with $sgn(\cdot)$. Besides, this operation can be understood by multi-stage pretraining, i.e. training the network with $tan(\beta_t \cdot)$ activation function is used to initialize the network with $tan(\beta_{t+1} \cdot)$ activation function. The two skills proposed by HashNet greatly increase the performance of deep supervised hashing.

Deep Priority Hashing[17] also adds different weights to different image pairs, but reduces the weight of pairs with higher confidence, which is similar to AdaBoost. The difficulty is measured by q_{ij}, which indicates how difficult a pair is classified as similar when $s_{ij}^o = 1$ or classified as dissimilar when $s_{ij}^o = 0$.

$$q\left(s_{ij}^o | \boldsymbol{h}_i, \boldsymbol{h}_j\right) = \begin{cases} \frac{1+s_{ij}^h}{2}, & s_{ij}^o = 1 \\ \frac{1-s_{ij}^h}{2}, & s_{ij}^o = 0 \end{cases}$$
$$= \left(\frac{1+s_{ij}^h}{2}\right)^{s_{ij}^o} \left(\frac{1-s_{ij}^h}{2}\right)^{1-s_{ij}^o}, \quad (19)$$

Besides, the weight characterizing class imbalance is measured by α_{ij}

$$\alpha_{ij} = \begin{cases} \frac{|\mathcal{S}_i||\mathcal{S}_j|}{\sqrt{|\mathcal{S}_i^1||\mathcal{S}_j^1|}}, & s_{ij} = 1 \\ \frac{|\mathcal{S}_i||\mathcal{S}_j|}{\sqrt{|\mathcal{S}_i^0||\mathcal{S}_j^0|}}, & s_{ij} = 0 \end{cases} \quad (20)$$

where $\mathcal{S}_i = \{(i,j) \in \mathcal{E} : \forall j\}$,

$$\mathcal{S}_i^1 = \left\{(i,j) \in \mathcal{E} : \forall j, s_{ij}^o = 1\right\}$$

and

$$\mathcal{S}_i^0 = \left\{(i,j) \in \mathcal{E} : \forall j, s_{ij}^o = 0\right\}$$

. The final priority weight is formulated as

$$w_{ij} = \alpha_{ij} \left(1 - q_{ij}\right)^\gamma,$$

where γ is a hyper-parameter. The weight for the pair with small confidence and balanced similar and dissimilar pairs will be large. Similarity, priority quantization loss changes the weight for different images to be $w_i' = (1 - q_i)\gamma$ and q_i measures how good a continuous code can be perfectly quantized into binary code. For these details, DPH achieved better performance than HashNet.

Deep Supervised Discrete Hashing[80] Besides likelihood based similarity information, DSDH also takes advantage of label information by adding a linear regression loss with regularization to loss function. By drop the binary restrictions, the loss is formulated as:

$$\mathcal{L} = - \sum_{(i,j) \in \mathcal{E}} \left(s_{ij}^o s_{ij}^h - \log\left(1 + e^{s_{ij}^h}\right)\right)$$
$$+ \eta \sum_{i=1}^n ||\mathbf{h}_i - sgn(\mathbf{h}_i)||_2^2 \quad (21)$$
$$+ \mu||\mathbf{y}_i - \mathbf{W}^T\mathbf{b}_i||$$
$$+ \alpha||\mathbf{W}||_F$$

where $s_{ij}^h = \frac{1}{2}\mathbf{h}_i^T\mathbf{h}_j$ and the label is encoded in one-hot format $\mathbf{y}_i$. the second term of the loss function is the linear regression term and the last term is an ℓ_2 regularization. $\mathbf{H}$, $\mathbf{B}$ and $\mathbf{W}$ are updated alternatively by using gradient descent method with discrete cyclic coordinate descend method. DSDH greatly increases the performance of image retrieval for it takes advantage of both label information and pairwise similarity information. It should be noted that in the linear regression term, the binary code is updated by discrete cyclic coordinate descend, so the constraint of discreteness is met.

Deep Cauchy Hashing[12] DCH propose a Bayesian learning framework just like DHN, but it replaced the exponential distribution by Cauchy distribution in the conditional probability. DCH aims to improve the search accuracy with Hamming distance smaller than 2 radius. Probability based on generalized sigmoid function can very large even for Hamming distance much larger than 2, which is harmful for Hamming ball retrieval. DCH addressed this problem by introducing Cauchy distribution, since the probability decrease very fast when Hamming distance larger than 2. The Cauchy distribution is formulated as

$$\sigma\left(d_{ij}^h\right) = \frac{\gamma}{\gamma + d_{ij}^h}, \quad (22)$$

in which γ is a hyper-parameter and the distance is the normalized Euclidean distance, i.e. $\mathbf{d}(\boldsymbol{h}_i, \boldsymbol{h}_j) = \frac{l}{2}(1 - cos(\boldsymbol{h}_i, \boldsymbol{h}_j))$. Besides, the prior is also based on a variant of the Cauchy distribution, i.e.

$$P\left(\boldsymbol{h}_i\right) = \frac{\gamma}{\gamma + \mathbf{d}\left(|\boldsymbol{h}_i|, \mathbf{1}\right)} \quad (23)$$

The final loss function is formulated as the log-likelihood plus the quantization loss based on the prior with weight. However, this loss function will get almost the same hash code for images with the same label. And the loss function for the dissimilar pairs was not considered.

Maximum-Margin Hamming Hashing[69] In view of the shortcomings of DCH, MMHH utilizes the t-Distribution which makes the loss function has different form for for similar and dissimilar pairs. And the total loss is the

weighted sum of two losses. Besides, a margin H is proposed to avoid producing the exact same hash codes. The Cauchy distribution in DCH is replaced by

$$\sigma\left(d_{ij}^h\right) = \begin{cases} \frac{1}{1+\max\left(0,d_{ij}^h-H\right)}, & s_{ij}^o = 1 \\ \frac{1}{1+\max\left(H,d_{ij}^h\right)}, & s_{ij}^o = 0 \end{cases} \quad (24)$$

The loss function is the weighted log-likelihood of conditional probability i.e.

$$\begin{aligned} \mathcal{L} &= \sum_{(i,j)\in\mathcal{E}} w_{ij}\left(s_{ij}^o\right)\log\left(1+\max\left(0,d_{ij}^h-\mathbf{H}\right)\right) \\ &+ \sum_{(i,j)\in\mathcal{E}} w_{ij}\left(1-s_{ij}^o\right)\log\left(1+\frac{1}{\max\left(\mathbf{H},d_{ij}^h\right)}\right) \quad (25) \\ &+ \lambda\sum_{i=1}^{n}||\mathbf{h}_i - sgn(\mathbf{h}_i)||_2^2 \end{aligned}$$

the last term is standard quantization loss. MMHH also proposed a semi-batch optimization process for alleviating the imbalance problem. Specifically, the codes of the training set are store as the augmented memory. And for a new epoch, the pairwise loss is calculated by the new codes computed in this epoch and their similar and dissimilar pairs are added into memory. In general, MMHH solved the shortcomings of DCH, which greatly improves the search performance.

Deep Fisher Hashing[82] DFH points out that the pairwise loss minimization is similar to Fisher's Linear discriminant, which maximizing the distance between inter-class images whilst minimizing the distance between the intra-class images. Its logistic loss function is similar to MMHH and the loss function can be formulated as:

$$\begin{aligned} \mathcal{L} &= \sum_{(i,j)\in\mathcal{E}} s_{ij}^o\log\left(1+e^{d_{ij}^h+m}\right) \\ &+ \sum_{(i,j)\in\mathcal{E}}\left(1-s_{ij}^o\right)\log\left(1+e^{-d_{ij}^h+m}\right) \quad (26) \\ &+ \lambda\sum_{i=1}^{n}||\mathbf{h}_i - sgn(\mathbf{h}_i)||_2^2, \end{aligned}$$

where m is a margin. Besides, the quantized center loss is also added to the loss function, which minimizes intra-class distances and maximize inter-class distances with binary hash codes of each image.

Deep Asymmetric Pairwise Hashing[104] Similar to DPSH, DAPH also adopted an asymmetric strategy. The difference is that DAPH uses two networks with different parameters for the database and queries. Besides, the bit independence, bit balance and quantization loss are added to the loss function just like SH-

BDNN. The loss function is optimized update the two neural networks alternatively.

Deep Attention-guided Hashing[128] DAgH adopted a two step framework just like CNNH, while it utilize neural networks to learn hash codes in both two steps. The loss function in the first step is the combination of the log-likelihood loss and the difference loss with margin. After obtaining the binary hash codes, step two utilizes binary point-wise cross entropy as the loss function. Besides, in step one, DAgH uses the architecture includes Fully Convolutional Network with attention module for obtaining accurate deep features.

Deep Joint Semantic-Embedding Hashing[79] DSEH is the first work to introduce LabNet in deep supervised hashing. It also adopted a two-step framework but with LabNet and ImgNet respectively. LabNet is a neural network designed to capture abundant semantic correlation with image pairs, which can help to guide the hashing code learning in step two. $\mathbf{f}_i^l = f(\mathbf{y}_i,\theta)$ denotes the embedding labels in which $\mathbf{y}_i$ is the labels in one-hot format and the output $sgn(\mathbf{f}_i^l)$ is the hash code. LabNet just replaces the input from images to their label and learns the hash codes from labels with general hashing loss function. In the step two, ImgNet utilize an asymmetric loss between the labeled feature in step one and the new obtained feature from ImageNet $\mathbf{f}_j^v$, i.e. $s_{ij}^h = \mathbf{f}_i^{lT}\mathbf{f}_j^v$ and binary cross-entropy loss just like DAgH[128]. DSEH fully makes use of the label information from the perspectives of both pairwise loss and cross-entropy loss, which can help to generate discriminative and similarity-preserving hash codes.

Asymmetric Deep Semantic Quantization[127] increases the performance by utilizing two ImgNets in minimizing the gap between the real-continuous features and the discrete binary codes, and the difference loss is also added.

Deep Anchor Graph Hashing[22] The anchor graph means utilizing 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. At first, it samples a number of anchors and builds an anchor graph between training samples and anchors. Then the loss function can be divided into two parts. The first part contains general pairwise likelihood loss and linear regression loss. The loss in the second part is calculated by the distances between training samples and anchors that in the same class, and both deep features and binary codes are used when computing the distances. the loss function includes the distance between deep features of training samples and binary codes of anchors which belong to the same class, besides general pairwise likelihood loss and linear regression loss. DAGH fully utilizes the remaining labeled data during mini-

batch training and helps to obtain efficient binary hash codes.

5 Multiwise similarity preserving

In this section, we will review the category of deep hashing algorithms that using multiwise similarity preserving loss function. Specifically, this kind of loss tries to preserve the similarity orders for more than two items that computed in input space and Hamming space. It should be pointed out that triplet loss is the most popular loss among them for its simpleness.

Deep Neural Network Hashing[74] DNNH utilizes a variant of the triplet ranking loss[98] to preserve the relative similarities of images. Specifically, given a triplet $(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)$ satisfies $s_{ij}^o > s_{ik}^o$, the ranking loss with margin is formulated as

$$\mathcal{L}(\mathbf{h}_i, \mathbf{h}_j, \mathbf{h}_k) = max(0, 1 + d_{ij}^h - d_{ij}^h) \qquad (27)$$

The loss encourages the binary code $\mathbf{b}_j$ to be closer to the $\mathbf{b}_i$ than $\mathbf{b}_k$. By replacing the Hamming distance with Euclidean distance, the loss function becomes convex

$$\mathcal{L}(\mathbf{h}_i, \mathbf{h}_j, \mathbf{h}_k) = max(0, 1 + ||\mathbf{h}_i - \mathbf{h}_j||_2^2 - ||\mathbf{h}_i - \mathbf{h}_k||_2^2) \qquad (28)$$

Besides, DNNH uses a sigmoid activation function followed by a piece-wise threshold function to encourage the output to be close to binary codes. The piece-wise threshold function is defined as

$$g(s) = \begin{cases} 0, & s < 0.5 - \epsilon \\ s, & 0.5 - \epsilon \leq s \leq 0.5 + \epsilon \\ 1, & s > 0.5 + \epsilon \end{cases} \qquad (29)$$

where ϵ is a small positive hyper-parameter. It is evident that most elements of the output will be exact 0 or 1 by using this piece-wise threshold function, thus introducing less quantization loss.

Deep Regularized Similarity Comparison Hashing[131] Besides the triplet loss, DRSCH also took advantage of pairwise information by introducing a difference loss as the regularization term. In addition, the bit weights is included when calculating the distance in Hamming space.

Deep Triplet Supervised Hashing[117] DTSH replaced the ranking loss by the negative log triplet label likelihood as

$$\mathcal{L}(\mathbf{h}_i, \mathbf{h}_j, \mathbf{h}_k) = log(1 + e^{s_{ij}^h - s_{ik}^h - \alpha}) - (s_{ij}^h - s_{ik}^h - \alpha), \qquad (30)$$

by considering the conditional probability[80], where α is a hyper-parameter.

Deep Semantic Ranking-based Hashing[135] DSRH proposed a surrogate loss based on triplet loss. Given query q and database $\{\mathbf{x}_i\}_{i=1}^n$, ranking $\{r_i\}_{i=1}^n$ in database is defined as the number of labels shared with the query.

$$\mathcal{L} = \sum_{i=1}^n \sum_{j:r_j < r_i} w(r_i, r_j) \delta max(0, \rho + d_{qi}^h - d_{qj}^h), \quad (31)$$

where δ and ρ are two hyper-parameters. And $w_{(r_i, r_j)}$ is the weight for the triplet that given by

$$\omega(r_i, r_j) = \frac{2^{r_i} - 2^{r_j}}{Z}$$

The form of weight comes from Normalized Discounted Cumulative Gains[61] score and Z is a normalization constant which can be omitted. Besides, bit balance and weight regularization are also added to the loss function. DSRH improved the performance of deep hashing by ranking list and surrogate loss, especially on multi-label image datasets.

6 Implicit similarity preserving

In this section, we will review deep hashing works whose loss functions are not exactly pair-wise or multi-wise similarity preserving. But they also take advantage of the similarity information, so we call this kind of loss implicit similarity preserving.

Hashing with Mutual Information[7] MIHash proposed a new loss following the idea of minimizing neighborhood ambiguity. This kind of loss is based on mutual information, which has direct and strong correlation with standard ranking-based retrieval performance metrics. Given y is a query image, the random variable $\mathcal{D}_{y,\Phi}$ is defined as a mapping from x to d_{xy}^h, where Φ is the hash function, i.e. deep neural network with binary output. $\mathcal{C}_y$ is the the set of images that share the same label with y, i.e. the neighbor of y. The mutual information is defined as

$$\mathcal{I}(\mathcal{D}_{y,\Phi}; \mathcal{C}_y) = H(\mathcal{C}_y) - H(\mathcal{C}_y|\mathcal{D}_{y,\Phi}) \qquad (32)$$

For any hash mapping Φ, the mutual information is integrated over the feature space to get a quality measure which desires to be maximized:

$$\mathcal{O} = -\int_\Omega \mathcal{I}(\mathcal{D}_{y,\Phi}; \mathcal{C}_y) p(y) dy, \qquad (33)$$

where Ω is the sample space and $p(y)$ denotes the prior distribution which can be removed. After discretion, the

loss function turns into:

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n}\mathcal{I}(\mathcal{D}_{x_i,\Phi};\mathcal{C}_{x_i}), \qquad (34)$$

whose gradient can be calculated by relaxing hard constraints and propagated by effect minibatch back propagation. The minibatch BP was proposed to effectively retrieving one example against the other example within a minibatch cyclically like leave-one-out validation.

Hashing as Tie-Aware Learning to Rank[49] HALR proposed to learn to rank formulations for hashing, which aimed at directly optimizing ranking-based evaluation metrics such as Average Precision (AP) and Normalized Discounted Cumulative Gain (NDCG). It is noticed that the integer-valued Hamming distance often leads to tied ranks. So HALR proposed a tie-aware versions of AP and NDCG, and derives their continuous relaxations as the loss function(plus sign) for optimization.

Angular Deep Supervised Hashing[136] Angular DSH calculates the Hamming distance between images of different classes to form a upper triangular matrix of G with size C by C. The mean of Hamming distance matrices needs to be maximized and the variance of the matrices needs to be minimized to ensure that these binary codes can cover all matrix elements and there is no short board in the bucket theory, i.e. bit balance. Besides, this method utilizes classification loss like PCDH but replaces the softmax loss by (dynamic) A-softmax[91] loss which can achieve larger inter-class separation but inter-class variation.

7 Classification-oriented deep hashing

In this section, we will review deep supervised hashing methods that take advantage of classification information instead of similarity information. For the hash codes are consistent with their labels, transfer learning and feature extraction are popular among these methods.

Deep Binary Hashing[84] After pre-training of a convolution neural network on the ImageNet to learn mid-level image representations, DBN adds a latent layer with sigmoid activation, the neurons in which is utilized to learn hash-like representation while fine-tuning with classification loss on the target domain dataset. The output in the latent layer is discretized into binary hash codes. DBN also emphasizes that the obtained hash codes is for coarse-level search because the quantity of hash codes is limited.

Supervised Semantics-preserving Deep Hashing[126] also utilizes the similar architecture as DBH but adds

regularization terms of quantization loss and bit balance.

Very Deep Supervised Hashing[134] proposed to utilize very deep neural network and train model with efficient algorithm layer-wise inspired by alternating direction method of multipliers(ADMM) [4].

SUBIC[60] generates structured binary hash codes consisiting of the concatenation of several one-hot encoded vectors(i.e. blocks) and obtains each one-hot encoded vector with several softmax function(i.e. block-softmax). Besides classification loss and bit balance regularization, the model utilizes the mean entropy for quantization loss for each block.

Mutual Linear Regression-based Discrete Hashing[92] utilizes the mutual linear regression for the loss function, i.e.

$$\mathcal{L} = ||\mathbf{Y}-\mathbf{W}^T\mathbf{B}||^2 + \alpha||\mathbf{B}-\mathbf{W}\mathbf{Y}||^2 + \lambda||\mathbf{W}||^2, \quad (35)$$

where $\mathbf{B}$ is the binary hash codes and $\mathbf{Y}$ is the label matrix. Similar to DSDH, the model is optimized by DCC approach. Besides, it also proposed a hash boosting step after obtaining several hash matrices according to bit balance.

Central Similarity Hashing[130] CSH also utilizes classification model but in a different way. First, CSH generates some central hash codes by the properities of a Hadamard matrix or random sampling from Bernoulli distributions, such that the distance between each pair of centroids are large enough. Each label is corresponding to a centroid and thus each images has its corresponding semantic hash center according to its label(single-label data). Afterwards, the model is trained by the central similarity loss(i.e. binary cross-entropy) with the supervised information(i.e.semantic hash center) plus the quantization loss. It is evident that CSH directly maps the predicted label to the corresponding centroid except adding some relaxations.

Hadamard Codebook Based Deep Hash[21] also utilizes Hadamard matrix by minimizing the ℓ_2 difference between hash-like output and the target hash codes with their corresponding labels(i.e. Hadamard loss) Different from CSH, HCDH trains the classification loss and Hadamard loss simultaneously. Hadamard loss can be interpreted as learning the hash centers guided by their supervised labels in L_2 norm. It's also noticed that HCDH is able to yield discriminative and balanced binary codes for the property of Hadamard codebook.

8 Quantization-based deep Hashing

It has been shown that the quantization approach can be derived from the difference loss minimization criterion[116].

Similar statement was proposed in [62] from the statistical perspective: the distance reconstruction error is statistically bounded by the quantization error. As a result, quantization can be used for deep supervised hashing. In this section, we will review the typical deep supervised hashing methods based on quantization.

Deep Quantization Network[14] DQN proposed to use product quantization approach to construct compact binary hash code b_i from the similarity-preserving bottleneck representation $z_i \in \mathcal{R}^R$. First, the original vector space is decomposed into the Cartesian product of M low-dimensional subspaces and each subspace is quantized into K codewords via clustering. Specifically, the original feature is partitioned into M sub-vectors, i.e.$\mathbf{z}_i = [\mathbf{z}_{i1}; \ldots; \mathbf{z}_{iM}], i = 1, \ldots, n$ and $\mathbf{z}_{im} \in \mathcal{R}^{R/M}$ is the sub-vector of $\mathbf{z}_i$ in the m-th subspace. Then all sub-vectors of each subspace are quantized into K clusters(codewords) independently through K-means as

$$Q = \sum_{m=1}^{M} \sum_{i=1}^{n} ||\mathbf{z}_{im} - \mathbf{C}_m \mathbf{b}_{im}||_2^2, \qquad (36)$$

$$||\mathbf{b}_{im}||_0 = 1, \mathbf{b}_{im} \in \{0,1\}^K,$$

where $\mathbf{C} = [\mathbf{c}_{m1}, \ldots, \mathbf{c}_{mK}]$ denotes the codebook of K codewords in the m-th subspace, and $\mathbf{b}_{im}$ is the code to indicate which codeword in $\mathbf{C}_m$ should be used to approximate the i-th point $\mathbf{z}_{im}$. Mathematically, product quantization can be reformulated as

$$Q = \sum_{i=1}^{n} ||\mathbf{z}_i - \mathbf{C}\mathbf{b}_i||_2^2, \qquad (37)$$

where $\mathbf{C}$ is a $R \times MK$ matrix can be written as

$$\mathbf{C} = diag(\mathbf{C}_1, \ldots, \mathbf{C}_M).$$

It's noticed that by minimizing Q, the quantization loss of converting the feature $\mathbf{z}_i$ into compact binary code $\mathbf{b}_i$ can be controlled. Besides, quantization-based hashing also adds pairwise similarity preserving loss to the final loss function. Finally, Asymmetric Quantizer Distance is widely used for approximate nearest neighbor search, which is formulated as

$$AQD(\mathbf{q}, \mathbf{x}_i) = \sum_{m=1}^{M} ||\mathbf{z}_{qm} - \mathbf{C}_m \mathbf{b}_{im}||_2^2, \qquad (38)$$

where $\mathbf{z}_{qm}$ is the m-th sub-vector for the feature of query $\mathbf{q}$.

Deep Triplet Quantization[87] uses triplet loss to preserve the similarity information and adds a weak orthogonality penalty across the M codebooks which is similar to the bit independence. And the orthogonality penalty can be formulated as

$$\sum_{m=1}^{M} \sum_{m'=1}^{M} ||\mathbf{C}_m^T \mathbf{C}_{m'} - \mathbf{I}||^2$$

. In addition, DTQ proposed to select triplet by Group Hard to ensure the number of mined valid triplets is neither too big nor too small. Specifically, the training data is split into several groups, and one hard (i.e. with positive triplet loss)negative sample is selected randomly for each anchor-positive pair in one group.

Deep Visual-semantic Quantization[13] learns deep quantization models from labeled image data as well as the semantic in- formation underlying general text domains. Specifically, it constructs deep visual-semantic embedding by taking the image representations $\mathbf{z}$ of the pre-trained convolutional neural network. And then retrain them to predict the word embeddings $\mathbf{v}$ (i.e. $\mathbf{v}_i$ for label i) of the image label-text, which was learned by the skip-gram model. The loss function includes the adaptive margin ranking loss

$$\mathcal{L} = \sum_{j \in \mathbf{y}_i} \sum_{k \notin \mathbf{y}_i} max(0, \delta_{jk} - cos(\mathbf{v}_j, \mathbf{z}_i) + cos(\mathbf{v}_k, \mathbf{z}_i)), \qquad (39)$$

where $\mathbf{y}_i$ is the label set of the i-th image, and δ_{jk} is a margin. Similarly, the quantization loss is inspired by the maximum inner-product search.

$$Q = \sum_{j=1}^{|\mathbf{y}|} (\mathbf{v}_j, \mathbf{z}_i - \mathbf{C}\mathbf{b}_i)^2. \qquad (40)$$

DVSQ adopted the same strategy with LabNet that discussed before and combines the visual information and semantic quantization in an uniform framework instead of two-step approach.

Deep Product Quantization[70] DPQ uses both the expressive power of PQ and the end-to-end learning ability of deep learning, making it possible to optimize the clustering results of PQ through classification tasks. Specifically, it first uses a embedding layer and a small multilayer perception to obtain the deep representation$z \in \mathcal{R}^{MN}$. Then the representation is sliced into M sub-vectors with $Z_m \in \mathcal{R}^N$ just like PQ. For each sub-vector, a small MLP are used to turn it into a probabilistic vector with K elements $p_m(k), k = 1, \ldots, K$ by softmax. Similar to PQ, The matrix $\mathbf{C}_m \in \mathcal{R}^{K \times D}$ denotes the K centroids. And $p_m(k)$ denotes the probability that the m-th sub-vector is quantized by the k-th row of $\mathbf{C}_m$(i.e. $\mathbf{C}_m(k)$). The soft representation of the m-th sub-vector is computed as the convex combination of the rows of $\mathbf{C}_m$.

$$soft_m = \sum_{k=1}^{K} p_m(k)\mathbf{C}_m(k).$$

Considering the probability $p_m(k)$ in one-hot format, given $k^* = argmax_k p_m(k)$, the hard probability is denoted as $e_m(k) = \delta_{kk*}$ in one-hot format and

$$hard_m = \sum_{k=1}^{K} e_m(k)\mathbf{C}_m(k).$$

The obtained sub-vectors of soft and hard representation are concatenated into the final representation $soft = [soft_1, \dots, soft_M]$ and $hard = [hard_1, \dots, hard_M] \in \mathcal{R}^{MD}$. Each representation is followed by a fully-connected classification layer. Beside two classification losses, the Joint Central Loss is also added by first learning the center vector for each class and minimizing the distance between the representations. It is noticed that DPQ uses the same centers for both the soft and hard representations, encouraging both representations to be closer to the same centers of the classes, thus decreasing the discrepancy between the soft and hard representations. This helps to improve the discriminative power of the features and contributing to the retrieval performance. Gini batch loss and Gini Sample loss are also introduced for the class balance and encouraging the two representations of the same image to be closer. Overall, DPQ replaces the k-means process in PQ technique with deep learning combined with classification model and is able to create a compressed representation for both fast classification and fast retrieval.

Deep Spherical Quantization[33] DSQ firset uses deep neural network to obtain the ℓ_2 normalized features. And then quantize the features using a new supervised quantization technique specifically designed for points lying on a unit hypersphere. After constraining the deep features to lie on a p-dimensional unit hypersphere(i.e. with standard norm), DSQ tries to minimize the distance reconstruction error of Multi-Codebook Quantization(MCQ). Different from PQ, MCQ approximates vectors with summation of multiple codewords instead of the concatenation. The quantization loss is formulated as

$$\mathcal{Q} = \sum_{i=1}^{n} ||\mathbf{z}_{im} - [\mathbf{C}_1, \dots, \mathbf{C}_M]\mathbf{b}_i||_2^2 \qquad (41)$$

$$||\mathbf{b}_{im}||_0 = 1, \mathbf{b}_i \in \{0,1\}^K, \mathbf{b}_i = [\mathbf{b}_{i,1}^T, \dots, \mathbf{b}_{i,M}^T]^T.$$

Besides the softmax loss, the discriminative loss and center loss are also included to encourage the quantized points and deep features to be closer to their centers, respectively.

Similarity Preserving Deep Asymmetric Quantization[20] adopts Asymmetric Quantizer Distance (AQD) to approximate the predefined similarity metric, which is similar to ADSH. Different from ADSH, it uses Composite Quantization instead of Product Quantization

and the representation in the training set comes from the deep neural network in an unquantized form. SP-DAQ also took advantage similarity information and label information to achieve better retrieval performance.

9 Other Topic in Deep Supervised Hashing

9.1 Learning to hash with Generative Adversarial Networks

Generative Adversarial Networks (GANs)[42] are powerful models for generating images in a minimax game mechanism without requiring supervised information.

Deep Semantic Hashing with GAN[101] is the first hashing method that took advantage of GANs for image retrieval. DSH-GANs consists of four components: a deep convolution neural networks (CNN) for learning image representations, an adversary stream to distinguish synthetic images from real ones, a hash stream for encoding image representations to hash codes and a classification stream. Specifically, for the generator network, it tries to synthesize realistic images with the concatenation input of the class label vector and random noise vector. For the discriminator network, it tries to simultaneously distinguish real images from synthetic ones and classify input images with correct class labels. The whole architecture is trained with the adversarial loss for assigning correct source and the classification loss for assigning correct class label in a two-player minimax game mechanism. The input is in the form of real-synthetic image triplets and each tuple consists of three images. The first one is real image that viewed as query image, the second one is synthetic and similar image produced with same labels of query image by generator network, and the third one is synthetic but dissimilar image. Besides data augmentation, GAN offers a hashing model with high generalization ability from the preserving of semantics and similarity.

HashGAN[11] augments the training data with images synthesized by Pair Conditional Wasserstein GAN inspired by [44] conditioned on the pairwise similarity information. PC-WGAN takes as the training images and pairwise similarity as inputs and jointly learns a generator and a discriminator by adding the pairwise similarity besides the loss function of WGAN. The hash encoder generates compact hash codes for both synthetic and real images in a Bayesian framework similar to that in HashNet. HashGAN is able to cope with the dataset without class labels but with similarity information.

9.2 Ensemble Learning

[45] pointed out that for the current deep supervised hashing model, simply increasing the length of the hash code cannot significantly improve the performance. Since the loss function adopted by existing methods are prone to produce highly correlated and redundant hash codes.

Ensemble-based Deep Supervised Hashing[45] proposed to use an ensemble learning strategy to solve this problem. Specifically, it trains a lot of deep hashing models with different training dataset and by training a lot of deep hashing models with different training data, initialization and networks, then concatenates them into the final hash codes. It is noticed that the ensemble strategy is suitable for parallelization and incremental learning.

Weighted Multi-deep Ranking Supervised hashing[78] tries to construct a more robust and discriminative learner with a set of deep hash tables. To be specific, WMRSH adds bit-wise weights and table-wise weights for each bit in each table(model). For each bit in a table, the similarity preservation is measured by product loss. Afterwards, the bit diversity is measured by the correlation between two hash bits. Finally, the Mean average precision (MAP) score for each hash table is viewed as the table-wise weight. The final weight is the product of three above terms for the final hash codes (i.e. the concatenation of the hash tables with weights). Similar strategy called Hash Boosting has been introduced before[92].

9.3 Training strategy for deep hashing

In this subsection, we will introduce two methods that adopted different training strategy from most other methods.

Greedy Hash[111] adopts greedy algorithm for fast processing of hashing discrete optimization by introducing a hash layer with sign function instead of the quantization error. And the the gradients are transmitted entirely to the front layer which effectively prevents the vanishing gradients of sign function and updates all bits together.

[56] points out a dilemma in learning deep hashing models through gradient descent that it makes no difference to loss if the paired hash codes change their signs together. As a result, *Gradient Attention Network[56]* trains a gradient attention network to generate attentions on the derivatives of each hash bit for each image by maximizing the decrease of loss. The attention weights generated by the gradient attention network with two fully-connected layers are normalized and then applied on the derivatives in the last layer. In conclusion, this model optimizes the training process by adopting a gradient attention network for effective learning.

9.4 Deep Unsupervised Hashing

Unsupervised methods do not require any label information. As a result, the similarity information are obtained by the distance of features. The unsupervised methods can be classified into three categories: the similarity-removed deep hashing, generative model based deep hashing and pseudo-label learning based deep hashing which converting the unsupervised problem into supervised problem. The models in the first category often come from the deep supervised hashing models but removing the supervised information.

Deep unsupervised hashing based on removing similarity information is very simple.[72] is the first deep unsupervised hashing methods to map images to binary codes by using auto-encoders. Several early deep supervised methods can be converted into deep unsupervised forms by removing the loss with semantic information [34,32]. DeepBit[83] uses the same framework with general Deep Supervised hashing methods like DSH. Since the similarity information and label information is unavailable, the model is trained by minimizing the quantization loss and bit balance regularization loss in this first period. Then the rotation invariant loss is added by augmenting the training data with different rotations to minimize the distance between binary code in the second period.

Unsupervised hashing based on generative models are limited for their complexity. Stochastic generative hashing[26] proposed to utilize generative model to learn hash functions through Minimum Description Length principle such that the learned hash codes maximally compress the dataset. And it can also used to regenerate the inputs. Unsupervised Deep Generative Adversarial Hashing Network[38] combines the hash encoder with Generative Adversarial Network by sharing the weights of the last layer with the discriminator. Besides, hash encoder has similar loss function with DeepBit.

The last strategy of unsupervised hashing is widely used and leads to better performance. Unsupervised learning of discriminative attributes and visual representation[54] also adopted a two-step framework. In the first stage, a CNN is trained coupled with unsupervised discriminative clustering[109]. In the second stage, the cluster membership is used as a soft supervision to discover shared attributes from the clusters while maximizing their separability in the form of triplet loss. In general, the unsupervised hashing is converted into super-

vised problem with similarity information from pseudo labels obtained before. Pseudo Label based Unsupervised Deep Discriminative Hashing[53] has the similar framework expect for using the classification loss instead of the similarity information. Unsupervised triplet hashing[57] constructs the image triplet by an anchor image, a rotated image and a random image. Afterward, the problem becomes a deep supervised hashing problem with triplet input. Unsupervised Deep Hashing with Similarity-Adaptive and Discrete Optimization[106] trains the model alternatively over three modules: deep hash model training, similarity graph updating and binary code optimization. It's noticed that in step one, the output in the step three is used for supervised information, the deep hash-like feature are utilized to construct data similarity graph, and then the binary code is obtained by solving the graph hashing problem. Different from [53, 109], this training strategy helps to improve the robustness of the model. Semantic Structure-based Unsupervised Deep Hashing[122] constructs the similarity matrix through the pairwise distances and then the model is trained by deep supervised hashing methods. DistillHash[124] improves the performance of SSUDH by using a different deep supervised hashing model and distilling the data pairs for confident similarity signals. Unsupervised Deep Hashing with Pseudo Labels[39] first obtains the pseudo labels embedding through maximum likelihood and maximum correlation orderly. Then the obtained pesudo labels are viewed as label information for the deep supervised hashing afterwards. Unsupervised Semantic-Preserving Adversarial Hashing[29] has the similar architecture to [38] but using the neighbor similarity obtained by K-nearest neighbor algorithm. Weakly Supervised Deep Image Hashing through Tag Embeddings[36] took advantage of user-generated tags associated with the images to learn the hash codes by constructing pairwise similarity information with tags.

9.5 Multi-modal Deep Hashing Methods

With the advent of the information age and the rapid development of the Internet, multimedia data has explosive growth in various modalities such as text, image, audio, and video. Multi-model deep hashing has arose much interest in the field of deep hashing recently. The framework of multi-modal deep hashing methods is similar to general deep hashing methods except that the similarity information includes the intra-modal and inter-modal forms. However, each loss term characterizing the similarity information is similar to that in deep supervised hashing discussed above. [9] gives a detailed review for the multi-modal hashing methods that includes [65, 123, 77, 133, 24, 63, 129, 43, 10, 31, 52].

10 Evaluation protocols

10.1 Evaluation Metrics

For deep hashing algorithms, the space cost only depends on the length of the hash code, so the length is usually kept the same when comparing the performance for different algorithms. The search efficiency is measured by the average search time for a query, which is mainly depends on the architecture of the neural networks. Besides, if the weighted Hamming distance is used, we cannot take advantage of bit operation for efficiency.

As discussed above, we usually use search accuracy to measure the performance. The most popular matrices include Mean Average Precision, Recall, Precision as well as the precision-recall curve. *Precision*: Precision is defined by the proportion of returned samples that share common label with the query. The formula can be formulated as:

$$precision = \frac{T}{T + N},$$

where T denotes the number of returned samples have common label with the query and N denotes the number of returned samples do not have common label with the query. $precision@k$ means the total number of returned sample is k, i.e. $T + N = k$.

Recall: Recall is define by the proportion of samples in the database that have common label with the query that are retrieved. The formula can be formulated as:

$$recall = \frac{T}{F},$$

where F is the total number of samples in the database that have common label with the query, including samples not retrieved. $recall@k$ means the total number of returned sample is k.

Precision-recall curve: In image retrieval, the precision rate and recall rate are dependent on k. The precision rate and recall rate of the same method are negatively correlated. Therefore, we can use the precision rate and recall rate as the horizontal and vertical coordinates to draw the precision-recall curve by varying k to further measure the performance.

Mean average precision: The average accuracy is calculated by integrating the precision rate against the x-axis when the recall rate changes from 0 to 1. In practical applications with discretion, the sequence summa-

tion method is utilized to calculated the average precision.

$$AP = \frac{1}{F} \sum_{k=1}^{n} precision@k\Delta\{T@k\},$$

in which $\Delta\{T@k\}$ denotes the change in recall from item $k-1$ to k. It's evident that the sum of $\Delta\{T@k\}$ is F. As a result, the core idea of AP is to evaluate a ranked list by averaging the precision at each position. Afterwards, MAP is the mean of the average precision of all query data. Some researches also calculate the MAP with Hamming Radius r, when only samples that have distances not bigger than r are considered.

10.2 Datasets

The widely-used evaluation datasets have different scales from small, large, to very large. The datasets can be categorized into single-label datasets and multi-label datasets.

MNIST[76] contains 60,000 training images and 10,000 testing images. Each image is described by 784-dimensional raw pixel features and 10K features as the queries.

CIFAR-10[71] contains 60,000 32×32 color images in 10 different classes.The 10 different classes represent airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. There are 6,000 images of each class. All of the images are identified with labels which can be used to evaluate the performance of hash based methods.

IMAGENET[30] is an large-scale image database contains more than 14 million images have been hand-annotated by the project to indicate what objects are pictured. ImageNet contains more than 20,000 categories with a typical category, such as "balloon" or "strawberry", consisting of several hundred images.

NUS-WIDE[25] is a real-world web image database from National University of Singapore. It includes 269,648 images with a total number of 5,018 unique tags. There are six types of low-level features extracted from these images: BoW, CH, CM55, CORR, WT and EDH. The images are manually assigned with some of the 81 concept tags. Since images are mostly associated with more than one label, one image is considered as the true nearest neighbor of the query if they contain at least one same label.

COCO[86] contains 82,783 training images and 40,504 validation images, each annotated by some of the 80 categories.

10.3 Performance Analysis

We present the results of the several representative deep supervised hashing and quantization algorithms over CIFAR-10 and NUS-WIDE. In CIFAR-10, 100 images are selected randomly per class (1,000 images in total) as the test query set, 500 images per class (5,000 images in total) as the training set. For NUS-WIDE, a subset of 195,834 images which are associated with the 21 most frequent concepts are selected. Each concept consists of at least 5,000 color images in this dataset. Afterwards, 100 images per class (2,100 images in total) are sampled randomly as the test query set, 500 images per class (10,500 images in total) as the training set.

It's noticed that for the various experimental settings, most of experimental results are not shown in this summary in detail. From the performance, here are some empirical results for different supervised hashing methods. (1)Deep supervised hashing greatly outperforms traditional hashing methods(e.g. SDH and KSH) overall. (2)Similarity information is necessary for deep hashing. For deep supervised hashing methods in early period(i.e. before 2016), hash codes are mostly obtained by transferring classification model without supervised similarity information while after the methods with pairwise or multi-wise information outperform them. (3)Label information helps to increase the performance of deep hashing. This point can be shown from the truth that DSDH outperforms DPSH evidently and the superiority of LabNet. What's more, some label-oriented methods[130, 21] show comparable performance recently. (4)Several skills including regularization term, bit balance, ensemble learning and bit independence help to obtain the accurate and robust performance. This point can be seen from ablation study in some paper.

11 Conclusion

In this article, we present a comprehensive review of the papers on deep hashing. Based on the similarity preserve manners, we We divide deep supervised hashing methods into five categories: pairwise similarity preserving, multiwise similarity preserving, implicit similarity preserving, quantization and classification oriented. In addition, we also introduce some latest topic such as learning to hashing with Generative Adversarial Network and multi-modal hashing methods. We observed that the existing deep hashing methods mainly focus on public dataset that designed for classification and detection, which do not fully address the nearest neighbor search problem. Future work is needed to combine downstream approximate nearest neighbor search algorithm to design specific deep hashing methods. And

Table 2 MAP for different methods on CIFAR-10 and NUS-WIDE datasets

Method	CIFAR-10				NUS-WIDE			
	12bits	24bits	32bits	48bits	12bits	24bits	32bits	48bits
CNNH[120]	0.439	0.511	0.509	0.522	0.611	0.618	0.625	0.608
DNNH[74]	0.552	0.566	0.558	0.581	0.674	0.697	0.713	0.715
DHN[137]	0.555	0.594	0.603	0.621	0.708	0.735	0.748	0.758
DTSH[117]	0.710	0.750	0.765	0.774	0.773	0.808	0.812	0.824
DPSH[81]	0.713	0.727	0.744	0.757	0.752	0.790	0.794	0.812
DSDH[80]	0.740	0.786	0.801	0.820	0.776	0.808	0.820	0.829
DQN[14]	0.554	0.558	0.564	0.580	0.768	0.776	0.783	0.792
SDH[105]	0.285	0.329	0.341	0.356	0.568	0.600	0.608	0.638
KSH[90]	0.303	0.337	0.346	0.356	0.556	0.572	0.581	0.588

then we can propose more practical deep hashing methods by this way.

References

1. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In: 2006 47th annual IEEE symposium on foundations of computer science (FOCS'06), pp. 459–468. IEEE (2006)
2. Beyer, K., Goldstein, J., Ramakrishnan, R., Shaft, U.: When is nearest neighbor meaningful? In: International conference on database theory, pp. 217–235. Springer (1999)
3. Böhm, C., Berchtold, S., Keim, D.A.: Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. ACM Computing Surveys (CSUR) **33**(3), 322–373 (2001)
4. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., et al.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends® in Machine learning **3**(1), 1–122 (2011)
5. Broder, A.Z.: On the resemblance and containment of documents. In: Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171), pp. 21–29. IEEE (1997)
6. Broder, A.Z., Glassman, S.C., Manasse, M.S., Zweig, G.: Syntactic clustering of the web. Computer networks and ISDN systems **29**(8-13), 1157–1166 (1997)
7. Cakir, F., He, K., Bargal, S.A., Sclaroff, S.: Hashing with mutual information. IEEE transactions on pattern analysis and machine intelligence **41**(10), 2424–2437 (2019)
8. Cakir, F., He, K., Sclaroff, S.: Hashing with binary matrix pursuit. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 332–348 (2018)
9. Cao, W., Feng, W., Lin, Q., Cao, G., He, Z.: A review of hashing methods for multimodal retrieval. IEEE Access (2020)
10. Cao, W., Lin, Q., He, Z., He, Z.: Hybrid representation learning for cross-modal retrieval. Neurocomputing **345**, 45–57 (2019)
11. Cao, Y., Liu, B., Long, M., Wang, J.: Hashgan: Deep learning to hash with pair conditional wasserstein gan. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1287–1296 (2018)
12. Cao, Y., Long, M., Liu, B., Wang, J.: Deep cauchy hashing for hamming space retrieval. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1229–1237 (2018)
13. Cao, Y., Long, M., Wang, J., Liu, S.: Deep visual-semantic quantization for efficient image retrieval. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1328–1337 (2017)
14. Cao, Y., Long, M., Wang, J., Zhu, H., Wen, Q.: Deep quantization network for efficient image retrieval. In: Thirtieth AAAI Conference on Artificial Intelligence (2016)
15. Cao, Y., Qi, H., Zhou, W., Kato, J., Li, K., Liu, X., Gui, J.: Binary hashing for approximate nearest neighbor search on big data: A survey. IEEE Access **6**, 2039–2054 (2017)
16. 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)
17. Cao, Z., Sun, Z., Long, M., Wang, J., Yu, P.S.: Deep priority hashing. In: Proceedings of the 26th ACM international conference on Multimedia, pp. 1653–1661 (2018)
18. Charikar, M.S.: Similarity estimation techniques from rounding algorithms. In: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing, pp. 380–388 (2002)
19. Chatfield, K., Simonyan, K., Vedaldi, A., Zisserman, A.: Return of the devil in the details: Delving deep into convolutional nets. arXiv preprint arXiv:1405.3531 (2014)
20. Chen, J., Cheung, W.K.: Similarity preserving deep asymmetric quantization for image retrieval. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 8183–8190 (2019)
21. Chen, S., Cao, L., Lin, M., Wang, Y., Sun, X., Wu, C., Qiu, J., Ji, R.: Hadamard codebook based deep hashing. arXiv preprint arXiv:1910.09182 (2019)
22. Chen, Y., Lai, Z., Ding, Y., Lin, K., Wong, W.K.: Deep supervised hashing with anchor graph. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 9796–9804 (2019)
23. Chen, Y., Lu, X.: Deep discrete hashing with pairwise correlation learning. Neurocomputing (2019)
24. Chen, Z., Zhong, F., Min, G., Leng, Y., Ying, Y.: Supervised intra-and inter-modality similarity preserving hashing for cross-modal retrieval. IEEE Access **6**, 27796–27808 (2018)
25. Chua, T.S., Tang, J., Hong, R., Li, H., Luo, Z., Zheng, Y.: Nus-wide: a real-world web image database from na-

tional university of singapore. In: Proceedings of the
ACM international conference on image and video re-
trieval, p. 48. ACM (2009)

26. Dai, B., Guo, R., Kumar, S., He, N., Song, L.: Stochastic
generative hashing. In: Proceedings of the 34th Interna-
tional Conference on Machine Learning-Volume 70, pp.
913–922. JMLR. org (2017)

27. Dasgupta, A., Kumar, R., Sarlós, T.: Fast locality-
sensitive hashing. In: Proceedings of the 17th ACM
SIGKDD international conference on Knowledge discov-
ery and data mining, pp. 1073–1081 (2011)

28. Datar, M., Immorlica, N., Indyk, P., Mirrokni, V.S.:
Locality-sensitive hashing scheme based on p-stable dis-
tributions. In: Proceedings of the twentieth annual sym-
posium on Computational geometry, pp. 253–262 (2004)

29. Deng, C., Yang, E., Liu, T., Li, J., Liu, W., Tao, D.: Un-
supervised semantic-preserving adversarial hashing for
image search. IEEE Transactions on Image Processing
28(8), 4032–4044 (2019)

30. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei,
L.: Imagenet: A large-scale hierarchical image database.
In: 2009 IEEE conference on computer vision and pat-
tern recognition, pp. 248–255. Ieee (2009)

31. Ding, K., Fan, B., Huo, C., Xiang, S., Pan, C.: Cross-
modal hashing via rank-order preserving. IEEE Trans-
actions on Multimedia **19**(3), 571–585 (2016)

32. Do, T.T., Doan, A.D., Cheung, N.M.: Learning to hash
with binary deep neural network. In: European Confer-
ence on Computer Vision, pp. 219–234. Springer (2016)

33. Eghbali, S., Tahvildari, L.: Deep spherical quantization
for image search. In: Proceedings of the IEEE Confer-
ence on Computer Vision and Pattern Recognition, pp.
11690–11699 (2019)

34. Erin Liong, V., Lu, J., Wang, G., Moulin, P., Zhou, J.:
Deep hashing for compact binary codes learning. In:
Proceedings of the IEEE conference on computer vision
and pattern recognition, pp. 2475–2483 (2015)

35. Friedman, J.H., Bentley, J.L., Finkel, R.A.: An algo-
rithm for finding best matches in logarithmic expected
time. ACM Transactions on Mathematical Software
(TOMS) **3**(3), 209–226 (1977)

36. Gattupalli, V., Zhuo, Y., Li, B.: Weakly supervised deep
image hashing through tag embeddings. In: Proceedings
of the IEEE Conference on Computer Vision and Pat-
tern Recognition, pp. 10375–10384 (2019)

37. Ge, T., He, K., Ke, Q., Sun, J.: Optimized product quan-
tization. IEEE transactions on pattern analysis and ma-
chine intelligence **36**(4), 744–755 (2013)

38. Ghasedi Dizaji, K., Zheng, F., Sadoughi, N., Yang, Y.,
Deng, C., Huang, H.: Unsupervised deep generative ad-
versarial hashing network. In: Proceedings of the IEEE
Conference on Computer Vision and Pattern Recogni-
tion, pp. 3664–3673 (2018)

39. Ghasedi Dizaji, K., Zheng, F., Sadoughi, N., Yang, Y.,
Deng, C., Huang, H.: Unsupervised deep generative ad-
versarial hashing network. In: Proceedings of the IEEE
Conference on Computer Vision and Pattern Recogni-
tion, pp. 3664–3673 (2018)

40. Gionis, A., Indyk, P., Motwani, R., et al.: Similarity
search in high dimensions via hashing. In: Vldb, vol. 99,
pp. 518–529 (1999)

41. Gong, Y., Lazebnik, S., Gordo, A., Perronnin, F.: It-
erative quantization: A procrustean approach to learn-
ing binary codes for large-scale image retrieval. IEEE
Transactions on Pattern Analysis and Machine Intelli-
gence **35**(12), 2916–2929 (2012)

42. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B.,
Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.:
Generative adversarial nets. In: Advances in neural in-
formation processing systems, pp. 2672–2680 (2014)

43. Gu, W., Gu, X., Gu, J., Li, B., Xiong, Z., Wang, W.:
Adversary guided asymmetric hashing for cross-modal
retrieval. In: Proceedings of the 2019 on International
Conference on Multimedia Retrieval, pp. 159–167 (2019)

44. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V.,
Courville, A.C.: Improved training of wasserstein gans.
In: Advances in neural information processing systems,
pp. 5767–5777 (2017)

45. Guo, Y., Zhao, X., Ding, G., Han, J.: On trivial solution
and high correlation problems in deep supervised hash-
ing. In: Thirty-Second AAAI Conference on Artificial
Intelligence (2018)

46. Hajebi, K., Abbasi-Yadkori, Y., Shahbazi, H., Zhang,
H.: Fast approximate nearest-neighbor search with k-
nearest neighbor graph. In: Twenty-Second Interna-
tional Joint Conference on Artificial Intelligence (2011)

47. Han, J., Zhang, D., Cheng, G., Liu, N., Xu, D.:
Advanced deep-learning techniques for salient and
category-specific object detection: a survey. IEEE Sig-
nal Processing Magazine **35**(1), 84–100 (2018)

48. He, J., Liu, W., Chang, S.F.: Scalable similarity search
with optimized kernel hashing. In: Proceedings of the
16th ACM SIGKDD international conference on Knowl-
edge discovery and data mining, pp. 1129–1138 (2010)

49. He, K., Cakir, F., Adel Bargal, S., Sclaroff, S.: Hash-
ing as tie-aware learning to rank. In: Proceedings of
the IEEE Conference on Computer Vision and Pattern
Recognition, pp. 4023–4032 (2018)

50. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learn-
ing for image recognition. In: Proceedings of the IEEE
conference on computer vision and pattern recognition,
pp. 770–778 (2016)

51. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learn-
ing for image recognition. In: Proceedings of the IEEE
conference on computer vision and pattern recognition,
pp. 770–778 (2016)

52. Hu, D., Nie, F., Li, X.: Deep binary reconstruction for
cross-modal hashing. IEEE Transactions on Multimedia
21(4), 973–985 (2018)

53. Hu, Q., Wu, J., Cheng, J., Wu, L., Lu, H.: Pseudo label
based unsupervised deep discriminative hashing for im-
age retrieval. In: Proceedings of the 25th ACM interna-
tional conference on Multimedia, pp. 1584–1590 (2017)

54. Huang, C., Change Loy, C., Tang, X.: Unsupervised
learning of discriminative attributes and visual repre-
sentations. In: Proceedings of the IEEE Conference on
Computer Vision and Pattern Recognition, pp. 5175–
5184 (2016)

55. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger,
K.Q.: Densely connected convolutional networks. In:
Proceedings of the IEEE conference on computer vision
and pattern recognition, pp. 4700–4708 (2017)

56. Huang, L.K., Chen, J., Pan, S.J.: Accelerate learning of
deep hashing with gradient attention. In: Proceedings of
the IEEE International Conference on Computer Vision,
pp. 5271–5280 (2019)

57. Huang, S., Xiong, Y., Zhang, Y., Wang, J.: Unsuper-
vised triplet hashing for fast image retrieval. In: Pro-
ceedings of the on Thematic Workshops of ACM Multi-
media 2017, pp. 84–92 (2017)

58. Hyvärinen, A., Hurri, J., Hoyer, P.O.: Natural image
statistics: A probabilistic approach to early computa-
tional vision., vol. 39. Springer Science & Business Me-
dia (2009)

59. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: Proceedings of the thirtieth annual ACM symposium on Theory of computing, pp. 604–613 (1998)

60. Jain, H., Zepeda, J., Pérez, P., Gribonval, R.: Subic: A supervised, structured binary code for image search. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 833–842 (2017)

61. Järvelin, K., Kekäläinen, J.: Ir evaluation methods for retrieving highly relevant documents. In: ACM SIGIR Forum, vol. 51, pp. 243–250. ACM New York, NY, USA (2017)

62. Jegou, H., Douze, M., Schmid, C.: Product quantization for nearest neighbor search. IEEE transactions on pattern analysis and machine intelligence **33**(1), 117–128 (2010)

63. Ji, Z., Yao, W., Wei, W., Song, H., Pi, H.: Deep multi-level semantic hashing for cross-modal retrieval. IEEE Access **7**, 23667–23674 (2019)

64. Jiang, Q.Y., Cui, X., Li, W.J.: Deep discrete supervised hashing. IEEE Transactions on Image Processing **27**(12), 5996–6009 (2018)

65. Jiang, Q.Y., Li, W.J.: Deep cross-modal hashing. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3232–3240 (2017)

66. Jiang, Q.Y., Li, W.J.: Asymmetric deep supervised hashing. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)

67. Jin, L., Shu, X., Li, K., Li, Z., Qi, G.J., Tang, J.: Deep ordinal hashing with spatial attention. IEEE Transactions on Image Processing **28**(5), 2173–2186 (2018)

68. Kalantidis, Y., Avrithis, Y.: Locally optimized product quantization for approximate nearest neighbor search. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2321–2328 (2014)

69. Kang, R., Cao, Y., Long, M., Wang, J., Yu, P.S.: Maximum-margin hamming hashing. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 8252–8261 (2019)

70. Klein, B., Wolf, L.: End-to-end supervised product quantization for image search and retrieval. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5041–5050 (2019)

71. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)

72. Krizhevsky, A., Hinton, G.E.: Using very deep autoencoders for content-based image retrieval. In: ESANN, vol. 1, p. 2. Citeseer (2011)

73. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp. 1097–1105 (2012)

74. 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)

75. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. nature **521**(7553), 436–444 (2015)

76. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)

77. Li, C., Deng, C., Li, N., Liu, W., Gao, X., Tao, D.: Self-supervised adversarial hashing networks for cross-modal retrieval. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4242–4251 (2018)

78. Li, J., Ng, W.W., Tian, X., Kwong, S., Wang, H.: Weighted multi-deep ranking supervised hashing for efficient image retrieval. International Journal of Machine Learning and Cybernetics pp. 1–15 (2019)

79. Li, N., Li, C., Deng, C., Liu, X., Gao, X.: Deep joint semantic-embedding hashing. In: IJCAI, pp. 2397–2403 (2018)

80. Li, Q., Sun, Z., He, R., Tan, T.: Deep supervised discrete hashing. In: Advances in neural information processing systems, pp. 2482–2491 (2017)

81. 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)

82. Li, Y., Pei, W., van Gemert, J., et al.: Push for quantization: Deep fisher hashing. arXiv preprint arXiv:1909.00206 (2019)

83. Lin, K., Lu, J., Chen, C.S., Zhou, J.: Learning compact binary descriptors with unsupervised deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1183–1192 (2016)

84. 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)

85. Lin, M., Chen, Q., Yan, S.: Network in network. arXiv preprint arXiv:1312.4400 (2013)

86. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European conference on computer vision, pp. 740–755. Springer (2014)

87. Liu, B., Cao, Y., Long, M., Wang, J., Wang, J.: Deep triplet quantization. In: Proceedings of the 26th ACM international conference on Multimedia, pp. 755–763 (2018)

88. Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.J., Fei-Fei, L., Yuille, A., Huang, J., Murphy, K.: Progressive neural architecture search. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 19–34 (2018)

89. Liu, H., Wang, R., Shan, S., Chen, X.: Deep supervised hashing for fast image retrieval. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2064–2072 (2016)

90. 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)

91. Liu, W., Wen, Y., Yu, Z., Li, M., Raj, B., Song, L.: Sphereface: Deep hypersphere embedding for face recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 212–220 (2017)

92. Liu, X., Nie, X., Yin, Y.: Mutual linear regression-based discrete hashing. arXiv preprint arXiv:1904.00744 (2019)

93. Lv, Q., Josephson, W., Wang, Z., Charikar, M., Li, K.: Multi-probe lsh: efficient indexing for high-dimensional similarity search. In: Proceedings of the 33rd international conference on Very large data bases, pp. 950–961 (2007)

94. Malkov, Y., Ponomarenko, A., Logvinov, A., Krylov, V.: Approximate nearest neighbor algorithm based on navigable small world graphs. Information Systems **45**, 61–68 (2014)

95. Malkov, Y.A., Yashunin, D.A.: Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. IEEE transactions on pattern analysis and machine intelligence (2018)

96. Motwani, R., Naor, A., Panigrahi, R.: Lower bounds on locality sensitive hashing. In: Proceedings of the twenty-second annual symposium on Computational geometry, pp. 154–157 (2006)

97. Muja, M., Lowe, D.G.: Fast approximate nearest neighbors with automatic algorithm configuration. VISAPP (1) **2**(331-340), 2 (2009)

98. Norouzi, M., Fleet, D.J., Salakhutdinov, R.R.: Hamming distance metric learning. In: Advances in neural information processing systems, pp. 1061–1069 (2012)

99. ODonnell, R., Wu, Y., Zhou, Y.: Optimal lower bounds for locality-sensitive hashing (except when q is tiny). ACM Transactions on Computation Theory (TOCT) **6**(1), 1–13 (2014)

100. Pham, H., Guan, M.Y., Zoph, B., Le, Q.V., Dean, J.: Efficient neural architecture search via parameter sharing. arXiv preprint arXiv:1802.03268 (2018)

101. Qiu, Z., Pan, Y., Yao, T., Mei, T.: Deep semantic hashing with generative adversarial networks. In: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 225–234 (2017)

102. Salakhutdinov, R., Hinton, G.: Semantic hashing. RBM **500**(3), 500 (2007)

103. Salakhutdinov, R., Hinton, G.: Semantic hashing. International Journal of Approximate Reasoning **50**(7), 969–978 (2009)

104. Shen, F., Gao, X., Liu, L., Yang, Y., Shen, H.T.: Deep asymmetric pairwise hashing. In: Proceedings of the 25th ACM international conference on Multimedia, pp. 1522–1530 (2017)

105. Shen, F., Shen, C., Liu, W., Tao Shen, H.: Supervised discrete hashing. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 37–45 (2015)

106. Shen, F., Xu, Y., Liu, L., Yang, Y., Huang, Z., Shen, H.T.: Unsupervised deep hashing with similarity-adaptive and discrete optimization. IEEE transactions on pattern analysis and machine intelligence **40**(12), 3034–3044 (2018)

107. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)

108. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)

109. Singh, S., Gupta, A., Efros, A.A.: Unsupervised discovery of mid-level discriminative patches. In: European Conference on Computer Vision, pp. 73–86. Springer (2012)

110. Strecha, C., Bronstein, A., Bronstein, M., Fua, P.: Ldahash: Improved matching with smaller descriptors. IEEE transactions on pattern analysis and machine intelligence **34**(1), 66–78 (2011)

111. Su, S., Zhang, C., Han, K., Tian, Y.: Greedy hash: Towards fast optimization for accurate hash coding in cnn. In: Advances in Neural Information Processing Systems, pp. 798–807 (2018)

112. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1–9 (2015)

113. Tan, M., Le, Q.V.: Efficientnet: Rethinking model scaling for convolutional neural networks. arXiv preprint arXiv:1905.11946 (2019)

114. Wang, J., Liu, W., Kumar, S., Chang, S.F.: Learning to hash for indexing big dataa survey. Proceedings of the IEEE **104**(1), 34–57 (2015)

115. Wang, J., Shen, H.T., Song, J., Ji, J.: Hashing for similarity search: A survey. arXiv preprint arXiv:1408.2927 (2014)

116. Wang, J., Zhang, T., Sebe, N., Shen, H.T., et al.: A survey on learning to hash. IEEE transactions on pattern analysis and machine intelligence **40**(4), 769–790 (2017)

117. Wang, X., Shi, Y., Kitani, K.M.: Deep supervised hashing with triplet labels. In: Asian conference on computer vision, pp. 70–84. Springer (2016)

118. Weiss, Y., Torralba, A., Fergus, R.: Spectral hashing. In: Advances in neural information processing systems, pp. 1753–1760 (2009)

119. Wu, D., Dai, Q., Liu, J., Li, B., Wang, W.: Deep incremental hashing network for efficient image retrieval. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 9069–9077 (2019)

120. Xia, R., Pan, Y., Lai, H., Liu, C., Yan, S.: Supervised hashing for image retrieval via image representation learning. In: Twenty-eighth AAAI conference on artificial intelligence (2014)

121. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1492–1500 (2017)

122. Yang, E., Deng, C., Liu, T., Liu, W., Tao, D.: Semantic structure-based unsupervised deep hashing. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence, pp. 1064–1070 (2018)

123. Yang, E., Deng, C., Liu, W., Liu, X., Tao, D., Gao, X.: Pairwise relationship guided deep hashing for cross-modal retrieval. In: Thirty-first AAAI conference on artificial intelligence (2017)

124. Yang, E., Liu, T., Deng, C., Liu, W., Tao, D.: Distillhash: Unsupervised deep hashing by distilling data pairs. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2946–2955 (2019)

125. 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 **1**(2), 3 (2015)

126. 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 **2** (2015)

127. Yang, Z., Raymond, O.I., Sun, W., Long, J.: Asymmetric deep semantic quantization for image retrieval. IEEE Access **7**, 72684–72695 (2019)

128. Yang, Z., Raymond, O.I., Sun, W., Long, J.: Deep attention-guided hashing. IEEE Access **7**, 11209–11221 (2019)

129. Yao, T., Zhang, Z., Yan, L., Yue, J., Tian, Q.: Discrete robust supervised hashing for cross-modal retrieval. IEEE Access **7**, 39806–39814 (2019)

130. Yuan, L., Wang, T., Zhang, X., Jie, Z., Tay, F.E., Feng, J.: Central similarity hashing via hadamard matrix. arXiv preprint arXiv:1908.00347 (2019)

131. Zhang, R., Lin, L., Zhang, R., Zuo, W., Zhang, L.: Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification. IEEE Transactions on Image Processing **24**(12), 4766–4779 (2015)

132. Zhang, T., Du, C., Wang, J.: Composite quantization for approximate nearest neighbor search. In: ICML, vol. 2, p. 3 (2014)
133. Zhang, X., Lai, H., Feng, J.: Attention-aware deep adversarial hashing for cross-modal retrieval. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 591–606 (2018)
134. 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)
135. 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)
136. Zhou, C., Po, L.M., Yuen, W.Y., Cheung, K.W., Xu, X., Lau, K.W., Zhao, Y., Liu, M., Wong, P.H.: Angular deep supervised hashing for image retrieval. IEEE Access **7**, 127521–127532 (2019)
137. Zhu, H., Long, M., Wang, J., Cao, Y.: Deep hashing network for efficient similarity retrieval. In: Thirtieth AAAI Conference on Artificial Intelligence (2016)
138. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578 (2016)