An Improved Similarity Algorithm Based On Deep Hash and Code Bit Independence

Jiyao Ding*, Anjun Chenga
School of Information Engineering, Wuhan University of Technology, Wuhan, China

*Corresponding author e-mail: 875281817@qq.com, 853722443@qq.com

Abstract. Deep convolutional neural networks have been widely used in image retrieval because of their powerful feature representation capabilities. Due to the high efficiency of hash space, many image retrieval algorithms based on deep hashing emerge. Aiming at the high correlation between the codes bits of hash code, this paper proposes a new end-to-end trainable deep hash algorithm to implement the feature grouping so that each code bit can express its unique partial image information. We add slice layers to reduce the repetitive expression of information and add the pairing loss to expand the difference between different categories of images and improve the recognition ability of the model for different categories. After training, we use the hierarchical search strategy to test the retrieval ability of the classification model. The mean average precision and recall of our algorithm can reach 87.5% and 92% respectively, which achieves great result in the related field and has important guiding significance for the future research of hash algorithm.

1. Introduction
Due to the accuracy of the nearest neighbor algorithm and the efficiency of the hash space, hashing has become an important technique for quickly approximating nearest neighbor search. Current hash-based methods can be subdivided into data-independent methods and data-dependent methods. Locally sensitive hashing (LSH) [1] is a typical data-independent method that randomly projects data sets without mining data structures, thus requiring longer code to achieve high recall accuracy. Data-dependent methods attempt to learn hash functions by using data distribution. Compared with data-independent methods, it is characterized by shorter code length and higher search accuracy. Therefore, data-dependent methods are becoming more and more common in both academic and industrial circles. The data dependence method can be further divided into a supervised method and unsupervised method. The difference between the two methods is whether they involve supervision information such as image class label. The spectral hash (SH) [2] and iterative quantization (ITQ) [3] are two representative unsupervised methods. The former obtains a balanced hash code by solving the spectrogram partition problem, and the latter approach is to maximize the variance of each bit and minimize the quantization loss that maps the data to the vertices of the binary hypercube to obtain the hash code. However, they all ignore the similarity between image semantics. It turns out that the supervised method has better search accuracy in many applications than unsupervised method. Supervised hash with kernel (KSH) [4] and supervised discrete hash (SDH) [5] are two representative methods of supervision. The former learns hash codes by continuously reducing the Hamming
distance between similar pairs and increasing the distance between relatives, and the latter uses the tag information to obtain the hash code by integrating the hash code and the generation of the classifier training. However, the traditional hash algorithm cannot overcome the problem of poor feature representation, so the retrieval accuracy is not effectively improved.

Recent advances in deep learning have shown that deep convolutional neural networks can learn rich visual descriptors. These descriptors illustrate excellent discriminability and bridge the gap between low-level pixels and high-level semantic information. Various deep learning algorithms have been widely used in visual analysis tasks, such as image classification, object detection, pedestrian detection [6], and face verification, showing their powerful feature extraction capabilities. Therefore, to learn image representation and hash code at the same time through the deep learning hash algorithm, the purpose of using the powerful feature representation ability from deep learning is to generate better feature vectors containing the original features of the image.

In this paper, we propose a new deep hash algorithm. In order to adapt to the hash learning task, we modify the structure of AlexNet's network and add a hash layer consisting of a slice layer, a fully connected layer, and a merge layer. Instead of using the activation function to control the output of the hash layer, we limit it to around +1 and -1 by quantization loss. The purpose is to reduce the quantization error caused by the conversion from Euclidean distance to Hamming distance. Reducing the similarity between the code points of the hash code can improve the retrieval accuracy. This theory has been proved in the literature [7]. The reason is that each code bit only expresses a specified partial image feature, which can reduce the overlap and repeated expression of image information. In this paper, we add a slice layer to split the input of the hash layer. The existence of pairing loss is to expand the difference between different categories of images while improving the recognition ability of the model for different categories. When we minimize these constraints, we can get the hash code we want.

2. Our Method
This section mainly introduces the proposed deep hash algorithm. The system framework is shown in Fig. 1.

As shown in Fig. 1, the input of the model is $X = \{I_1, I_2, \ldots, I_n\}$. $I_n$ Indicates the input n-th image data, and the corresponding label is $Y = \{y \in \{0,1\}^M\}$, where $M$ represents the total number of data set categories. $y = 1$ When $I_n$ belongs to the corresponding class, otherwise it is 0. Through the
trained network model, each image can be matched to its hash code $H = \{h_k\} \in \{-1, 1\}^{K \times N}$. $K$ represents the length of the hash code. The goal of our algorithm is to learn this mapping and preserve the original image information of $I$ in $H$.

2.1. Model Structure

Our network is modified appropriately based on the basic framework of the AlexNet network, adding a hash layer paralleled to the classification layer. The parameters of the new network are shown in Table 1.

| Layer     | Size/Stride | Filters Number | Others          |
|-----------|-------------|----------------|-----------------|
| Conv1     | 3x3/1       | 24             | padding=SAME   |
| Max pooling | 2x2/2       | None           | None            |
| Conv2     | 3x3/1       | 96             | padding=SAME   |
| Max pooling | 2x2/2       | None           | None            |
| Conv3     | 3x3/1       | 192            | padding=SAME   |
| Conv4     | 3x3/1       | 192            | padding=SAME   |
| Conv5     | 3x3/1       | 384            | padding=SAME   |
| FC6       | None        | 2048           | dropout=0.5     |
| FC7       | None        | 2048           | dropout=0.5     |
| H         | None        | $K$            | None            |
| Classification | None     | M              | None            |

As shown in Table 1, it has five convolutional (CONV) layers, two maximum pooling operations, followed by two fully connected (FC) layers, one hash layer and one classification layer. The ReLU function exists as an activation function in every CONV layer and FC layer, adding nonlinear attributes to the network and also speeding up training.

After training the model, we save two effective features of each image: hash code and feature vector (high-level semantic feature). The hash code can be obtained from the output of the hash layer, and the feature vector is the output of the second fully connected layer.

2.2. Hash Function

In reference [7], it propose that high quality hash codes should be as irrelevant as possible between code points. This is because each code bit only expresses a specified partial image feature, which can reduce overlap and repeated expression of the image information. Therefore, the retrieval accuracy of the image can be improved. Considering that the dimension of the hash code is $K$, we divide the input $a^F_i$ of the hash layer into an average of $K$ sub-chains $\{a^F_1, a^F_2, \cdots, a^F_K\}$. Each code bit is only allowed to be connected to one of the sub-chains. $W^H = \{W^H_1, W^H_2, \cdots, W^H_K\}$ represents the weight of the fully connected layer in the hash layer, where $W^H_i \quad i \in \{1,2,\cdots,K\}$ represents the weight connected to the i-th sub-chain. We can get the output of the i-th code bit of the quasi-hash code $a^H_i = a^F_i \cdot W^H_i + b^H_i$, among them $b^H_i$ is the i-th deviation term. Thus we can get all the code point values of the entire quasi-hash code $\{a^H_1, a^H_2, \cdots, a^H_K\}$, and generate a quasi-hash code $a^H$ through the merged layer. Finally, the final binary hash code is obtained by the symbol function.

It can be seen that the threshold function is not set in the hash layer of our model because the too small edge slope will reduce the training speed, even causing the gradient to disappear. In order to reduce the influence of excessive loss from quasi-Hash code to hash code conversion, we use quantization loss as one of the network optimization goals.
2.3. Loss Function

The loss function is one of the most important parts of the whole network model. As the factor of artificial design, its quality affects the whole training process of the model and determines the final retrieval performance. The model is continuously trained to reduce the loss to an acceptable range, resulting in a strong generalization capability. So we carefully design the loss function used by the model.

Our main goal is to learn the mapping from input image \( I_n \) to hash code \( h_n \) so that similar images get similar hash codes while ensuring independence between each hash code bit. Due to the lack of the activation function, the output of the hash layer has a large floating range, which is unfavorable for the training and performance evaluation of the network. Therefore, it is necessary to add a constraint to control the output range, which is used to reduce the quantization loss from the Euclidean space to the Hamming space while retaining the original data. Finally, as an important monitoring information, image tags can effectively improve the model classification ability and generate high-quality hash codes. Based on the above considerations, our loss function for the entire network can be expressed as:

\[
L = L_C + L_Q + L_P + \lambda \|W\|^2
\]  

Among them \( L_C \) indicates classification loss, \( L_Q \) representing quantization loss, \( L_P \) indicates the pairing loss. \( W \) is the weight of the entire network, and the last one can play a role in reducing the model overfitting problem. \( \lambda \) Indicates the relative importance of regularizations.

1) Classification loss

Hypothesis \( W^C \in R^{D \times M} \) represents the weight between the second fully connected layer and the classification layer, where \( D \) represents the output dimension of the FC7 layer. Vector \( \hat{y}_n \) indicates the output of the classification layer, which is also the prediction label of the input image \( I_n \). The classification loss can be expressed as:

\[
L_C = \frac{1}{N} \sum_{n=1}^{N} L(y_n, \hat{y}_n)
\]  

Where \( L(\cdot) \) represents the loss function that minimizes the classification loss. \( N \) Represents the number of input images for a batch. We use the softmax output and the cross entropy loss function, so:

\[
L(y_n, \hat{y}_n) = -\sum_{m=1}^{M} y_{nm} \cdot \ln \hat{y}_{nm}
\]  

Where \( y_{nm} \) represents the m-th element value of the real label vector of image \( I_n \) and \( \hat{y}_{nm} \) indicates the same meaning of the predicted image label.

2) Quantization loss

Since the output of the hash layer lacks the activation function and is relaxed to a real value, a quantization loss will be generated in the process of mapping the output from the Euclidean distance to the Hamming distance hash code, affecting the model accuracy. After obtaining the quasi-hash code \( a^h \), a quantization step can be defined as:

\[
h = \text{sign}(a^h)
\]  

Where \( \text{sign}(\cdot) \) is the sign function of the output. \( \text{sign}(a^h(k)) = 1 \) if \( a^h(k) > 0 \), otherwise -1, for \( k \in \{1,2,\cdots,K\} \).
In order to overcome the quantization loss from the Euclidean space to the Hamming space and retain the information of the original data, the quantization loss function is proposed, which can be written as:

$$L_0 = \frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} ||h_n(k) - a_n(k)||^2$$  \hspace{1cm} (5)

Where $h_n(k)$ represents the k-th hash code of the n-th input image. We calculate the difference between the original output $a_n$ and the quantized result $h$.

3) Pairing loss

Through reference [8], we randomly select two from a batch of $N$ images imported to get $\binom{N}{2}$ pairs. The advantage is that each set of image pairs in a batch of images can be considered, adjusted according to similarity. So that similar images output similar hash codes and vice versa. $S$ Indicates whether the pair of images are similar to each other. $S = 1$ If they are similar, otherwise 0.

For a single-label image set, we think that two images with the same label are semantically similar to each other. For multi-label data sets, because of the particularity of images sharing multiple tag attributes. We define that if two images share at least one tag, they are considered to be semantically similar, otherwise they are considered to be dissimilar. Therefore, $S$ can be defined as:

$$S = \begin{cases} 1 & \text{label}_1 = \text{label}_2 \\ 0 & \text{label}_1 \neq \text{label}_2 \end{cases}$$  \hspace{1cm} (6)

The former corresponds to a single-label image set and the latter corresponds to a multi-label image set.

We use pairing loss as one of the optimization items of network training. The pairing loss of a pair can be expressed as:

$$L_p(h_1, h_2) = \begin{cases} ||h_1 - h_2||^2 & S = 1 \\ \max(t - ||h_1 - h_2||^2, 0) & S = 0 \end{cases}$$  \hspace{1cm} (7)

Where $h$ represents a binary hash code. $t$ Represents the threshold and is an artificially defined parameter. It represents the minimum distance between different pairs of images that can be accepted. If this distance is exceeded, the model can assume that they are already sufficiently different, so the contribution to the loss function is zero. Therefore, the pairing loss can be finally expressed as:

$$L_p = \frac{1}{C_N^2} \sum_{i=1}^{N} \sum_{j=i+1}^{N} (S||h_i - h_j||^2 + (1 - S) \max(t - ||h_i - h_j||^2, 0))$$  \hspace{1cm} (8)

2.4. Model Retrieval Strategy

In order to verify the retrieval ability of the model, we use a rough-to-fine hierarchical search strategy. First, we save the hash code and feature vector (high-level semantic feature) corresponding to each image in the image library through the trained model. During the retrieval process, the user selects a
test set image as the model input query image $I_q$. We can get the corresponding hash code $h_i = \{h_1(1), h_2(2) \cdots h_K(K)\}$ and eigenvectors $V_r \in \mathbb{R}^{l \times D}$, $D$ represents the dimension of the feature vector column. In the rough retrieval stage, the efficiency of Hamming distance calculation can be used to quickly locate the query image. The formula is as follows:

$$d(h_i, h_j) = \sum h_i \oplus h_j \quad i \in [1, 2 \cdots X_{\text{train}}] \quad (9)$$

Where $h$ represents the binary hash code of the i-th image in the image library. $X_{\text{train}}$ Represents the number of images in the image library, $\oplus$ means XOR and $P$ images with a Hamming distance of 0 are reserved to form an image pool.

In the fine retrieval phase, we use the eigenvectors to calculate the Euclidean distance between the query image and all images in the image pool. The formula is as follows:

$$\text{Dist}_i = \|V_r - V_i\| \quad i \in [1, 2 \cdots P] \quad (10)$$

Where $V_i$ represents the feature vector of the i-th image in the image pool. After the retrieval is completed, the Euclidean distance is reordered from small to large and the first 8 images are returned to the user.

3. Experimental Results

In this section, we use our proposed algorithm to test on two common data sets and compare them to other traditional hashing algorithms and deep hashing algorithms. We use the mean average precision (mAP) and k sample accuracy (prep@k) as the indicator of performance comparison.

3.1. Experimental Setup

In order to test the retrieval accuracy of our algorithm, we select a single-label image set and a multi-label image set to test the performance of the model. CIFAR-10 consists of 10 categories of 60,000 32 x 32 single-label color images, of which 50,000 are training sets and 10,000 are test sets. NUS-WIDE covers nearly 270,000 images, each associated with one or more of the 81 semantic concepts. We extract images associated with the 21 most commonly used tags, each containing at least 5,000 images, and the total number of images is 195,834.

We compare the retrieval performance of our model with other methods. First, all comparison algorithms can be roughly divided into two parts: the traditional hash algorithm and the deep hash algorithm, where the traditional hash algorithm is subdivided into supervised and unsupervised. We choose unsupervised LSH [1] and supervised KSH [4]. About deep hash algorithm, we pick CNNH [9], DHN [10], DSH [11], DCNNH [12] and the algorithm in reference [7]. In addition, for all comparison methods, we use the experimental data from the original paper or reference [13] as a reference.

In CIFAR-10, we select all the training images as the training set. For the test set, we randomly select 1000 images (100 per class) from 10,000 images as the query images for performance evaluation. In NUS-WIDE, we randomly extract 2,100 (100 per class) from 21 classes as test sets and the remaining images are used as the training set of the model. Before starting the training, we adjust the image size to 256 * 256 pixels.

We use TensorFlow to build the model and train the network using a stochastic gradient descent algorithm with a momentum of 0.9. The batch size is 100 and the weight attenuation parameter $\lambda$ is 0.004. The threshold $t$ is set to 4, which can ensure that each type of binary hash code has at least one different code bit to achieve the classification accuracy.
3.2. Evaluation Criteria

We use the two most widely used evaluation indicators (mean average precision and k sample accuracy) to evaluate the performance of the model.

1) Mean average precision (mAP)

mAP is an overall evaluation indicator of retrieval performance. It is the average of the average precision (AP) of each query image, where AP can be calculated as:

\[
AP = \frac{1}{R} \sum_{k=1}^{R} R_k \times rel_k
\]  

(11)

Where \( X_{train} \) denotes the size of the training set, \( R \) is the number of related images in the data set, \( R_k \) is the number of related images in the first \( k \) returned images, \( rel_k \) represents the correlation between the returned \( k \)-th image and the query image. \( rel_k = 1 \) if relevant, otherwise 0.

2) \( k \) sample accuracy (prep@k)

prep@k represents the proportion of images in the first \( k \) images returned that are similar to the query image. Its formula can be expressed as:

\[
prep@k = \frac{1}{X_{test}} \sum_{i=1}^{k} \sum_{j=1}^{k} rel_{i,j}
\]  

(12)

Where \( X_{test} \) indicates the size of the test set, \( j \) represents the j-th image in the \( k \) returned images. \( rel_{i,j} \) indicates whether the j-th return image at the time of the i-th query is related to the query image. \( rel_{i,j} = 1 \) if relevant, otherwise 0.

3.3. Results on CIFAR-10

Table 2 shows the mAP values for the individual code lengths of the different algorithms on the CIFAR-10 data set and the best search results are highlighted in bold.

| Algorithm   | 12bits | 24bits | 32bits | 48bits |
|-------------|--------|--------|--------|--------|
| Ours        | 0.843  | 0.849  | 0.869  | 0.875  |
| literature[7]| 0.839  | 0.845  | 0.851  | 0.856  |
| DCNNH[12]   | 0.816  | 0.826  | 0.829  | 0.832  |
| DSH[11]     | 0.644  | 0.742  | 0.770  | 0.799  |
| DHN[10]     | 0.555  | 0.594  | 0.603  | 0.621  |
| CNNH[9]     | 0.439  | 0.511  | 0.509  | 0.522  |
| KSH[4]      | 0.303  | 0.337  | 0.346  | 0.356  |
| LSH[1]      | 0.121  | 0.126  | 0.120  | 0.120  |

As shown in Table 2, the retrieval precision of our algorithm increases with the increase of the code bit length. The reason is that when the total amount of information is constant, the longer the code position, the lower the unit information carrying capacity. Thus more valid information can be displayed through more code points. Apart from this, we can also find that the mean average precision of our proposed algorithm is always slightly higher than other comparison algorithms. Compared with the algorithm DCNNH, we consider the quantization loss and the independence of the hash code to
eliminate the dependence between the code bits of the hash code. So the performance is improved by 2.2\% - 4.3\%. Compared with the suboptimal algorithm in reference [7], we add the pairing loss in the loss layer which can improve the model's ability to distinguish different categories. The mean average precision of the model is improved by 0.4\% - 1.9\%.

We also count the k-sample precision prep@5000 of the first 5000 with different code lengths of different algorithms and draw a line graph as shown in Fig. 2.

![Figure 2. prep@5000 on CIFAR-10.](image)

As shown in Fig. 2, our algorithm has a slightly higher index of each code bit length than the closest control algorithm. Since the number of samples per class is 5000, prep@5000 indicates the recall rate. It can be seen that the recall rate of our algorithm is always higher than 90\% and increases with the increase of code length.

We use the trained 48-bit hash code network model to verify the retrieval effect of the model. We randomly select two different types of query images from the test set and specify that the image with a Hamming distance of 0 entered the image pool. The query results are shown in Fig. 3. The previous line represents 8 returned images which are randomly selected from the image pool. The next row represents the refine search of the image in the image pool by the feature vector and return 8 images from small to large according to the Euclidean distance.

![Figure 3. Hierarchical retrieval results of 48-bit hash code on CIFAR-10.](image)
As shown in Fig. 3, the rough search is biased to retrieve the same type of related image. But the difference in appearance such as shape and color are still relatively large. After retrieving and reordering the images in the image pool. The first 8 images returned can be semantically similar to the query image and the appearance will be closer to it.

3.4. Results on NUS-WIDE

To verify the applicability of the algorithm, we choose a multi-label image set. Table 3 shows the mAP values for different code lengths of different algorithms on the NUS-WIDE data set. We mark the best search results in bold.

| Algorithm  | 12bits | 24bits | 32bits | 48bits |
|------------|--------|--------|--------|--------|
| Ours       | 0.782  | 0.788  | 0.791  | 0.793  |
| literature [7] | 0.761 | 0.767 | 0.771 | 0.773 |
| DCNNH [12] | 0.741  | 0.748  | 0.752  | 0.753  |
| DSH [11]   | 0.712  | 0.731  | 0.740  | 0.748  |
| DHN [10]   | 0.708  | 0.735  | 0.748  | 0.758  |
| CNNH [9]   | 0.611  | 0.618  | 0.625  | 0.608  |
| KSH [4]    | 0.556  | 0.572  | 0.581  | 0.588  |
| LSH [1]    | 0.403  | 0.421  | 0.426  | 0.441  |

As shown in Table 3, the difference between the traditional hash and the deep hash algorithm is reduced. The reason is that each image of the multi-label image set shares multiple tags at the same time. The average sample accuracy of our algorithm can reach up to 79%, which is 2% higher than the suboptimal algorithm. It means that similar images appear earlier in the returned image than other algorithms and the retrieval effect of our algorithm is better.

We calculate the k-sample precision prep@5000 of the first 5000 different code lengths of each algorithm and plot the line diagram as shown in Fig. 4.

![Figure 4. prep@5000 on NUS-WIDE.](image)
As shown in Fig. 4, the sample accuracy of our method increases with the increase of the code length in the first three code lengths. It reaches the highest 81% over the length of 32 code bits. Although there is no strict limit on the number of samples per class in this data set. We can still approximate the accuracy of the sample as the recall rate. Therefore, we can see that the recall rate of our algorithm is always the highest. That is to say in a certain number of returned images, our algorithm can obtain more similar images to the query image and get better retrieval results.

4. Conclusion
In this paper, we propose a new deep hash algorithm by designing a new network model. We split the input of the hash layer so that each code point can express its unique partial image features. Therefore, the overlap and repeated expression of information are reduced, and the accuracy of the retrieval is indeed improved by experiments. In view of the poor ability of the model to classify different classes, we add pairing losses to expand the difference between different types of images. We design a series of loss functions based on the model requirements, and obtain similar and independent binary hash codes by continuously training to minimize our defined loss terms. The experimental results show that our algorithm achieves good results on both single-label image sets and multi-label image sets. The mean average precision and recall of our algorithm can reach 87.5% and 92% respectively, which is a great level in the related field. However, our algorithm has not considered the influence of low-level features on hash codes, so we will improve in future research.

5. References
[1] M Datar, N Immorlica, P Indyk et al. Locality-sensitive hashing scheme based on p-stable distributions. Proceedings of the Twentieth Annual Symposium on Computational Geometry, 2004, 253–262.
[2] Y Weiss, A Torralba and R Fergus. Spectral hashing. Proc Adv Neural Inf Process Syst, 2009, 1753–1760.
[3] Y Gong and S Lazebnik. A procrustean approach to learning binary codes. Comput Vis Pattern Recognit (CVPR), 2011, 817–824.
[4] W Liu, J Wang, R Ji, Y Jiang and S Chang. Supervised hashing with kernels. Comput Vis Pattern Recognit (CVPR), 2012, 2074–2081.
[5] F Shen, C Shen, W Liu and H Shen. Supervised discrete hashing. Comput Vis Pattern Recognit (CVPR), 2015, 37–45.
[6] J Li, X Liang, S Shen, T Xu, J Feng, S Yan. Scale-aware fast r-cnn for pedestrian detection. IEEE Transactions on Multimedia, 2018, 20(4): 985–996.
[7] T Peng and F Li. Image retrieval based on deep Convolutional Neural Networks and binary hashing learning. 2017 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP), 2017, 1742–1746.
[8] A Jose, S Yan and I Heisterklaus. Binary hashing using siamese neural networks. 2017 IEEE International Conference on Image Processing (ICIP), 2017, 2916–2920.
[9] R Xia, Y Pan, H Lai, et al. Supervised hashing for image retrieval via image representation learning. The National Conference on Artificial Intelligence, 2014, 1: 2156–2162.
[10] H Zhu, M Long, J Wang et al. Deep hashing network for efficient similarity retrieval. The National Conference on Artificial Intelligence, 2016, 2415–2421.
[11] H Liu, R Wang, S Shan et al. Deep supervised hashing for fast image retrieval. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, 2064–2072.
[12] K Lin, H Yang, J Hsiao and C Chen. Deep Learning of Binary Hash Codes for Fast Image Retrieval. Computer Vision and Pattern Recognition, 2015, 27-35.
[13] H Zheng, R Ma, P An and T Li. Feature-Selecting Based Hashing via Deep Convolutional Neural Networks. Communications in Computer and Information Science, 2019, 1009: 146–155.