

Supervised Hashing for Multi-labeled Data with Order-Preserving Feature

Dan Wang, Heyan Huang[✉], Hua-Kang Lin, and Xian-Ling Mao

Beijing Institute of Technology, Beijing, China
{wangdan12856,hhy63,1120141916,maoxl}@bit.edu.cn

Abstract. Approximate Nearest Neighbors (ANN) Search has attracted much attention in recent years. Hashing is a promising way for ANN which has been widely used in large-scale image retrieval tasks. However, most of the existing hashing methods are designed for single-labeled data. On multi-labeled data, those hashing methods take two images as similar if they share at least one common label. But this way cannot preserve the order relations in multi-labeled data. Meanwhile, most hashing methods are based on hand-crafted features which are costing. To solve the two problems above, we proposed a novel supervised hashing method to perform hash codes learning for multi-labeled data. In particular, we firstly extract the order-preserving data features through deep convolutional neural network. Secondly, the order-preserving features would be used for learning hash codes. Extensive experiments on two real-world public datasets show that the proposed method outperforms state-of-the-art baselines in the image retrieval tasks.

Keywords: Order-preserving feature · Supervised hashing · Multi-labeled data

1 Introduction

Approximate nearest neighbor (ANN) [1, 2] search plays a fundamental role in machine learning and related areas, such as image retrieval, pattern recognition and computer vision [3, 22, 25].

Hashing is an effective technology to solve ANN problems. It has received more and more attention in the big data era because of its fast retrieval speed and low storage cost. Hashing maps the data points from the original feature space into Hamming space to generate compact binary/hash codes. The more similar two points are in the original feature space, the smaller their hamming distance is. On the contrary, when two data points are dissimilar, a high hamming distance is expected between their hash codes.

Generally speaking, hashing methods can be divided into data-independent methods and data-dependent methods. Compared with data-independent hashing methods like Locality Sensitive Hashing (LSH) [5], data-dependent hashing

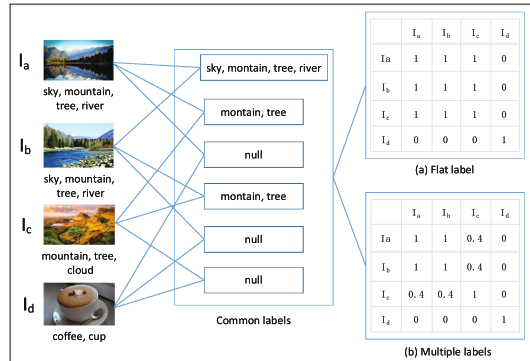


Fig. 1. Example of multi-labeled data. The left panel shows four images and their labels. The middle panel presents the common labels between images. The traditional similarities are listed in the upper-right panel (a): two images are similar when they share at least one label. The bottom-right panel (b) is the similarities considering the numbers of common labels. The more common labels two images have, the more similar they are.

methods can achieve comparable or better accuracy with shorter codes by utilizing the train data [6, 16, 17, 23]. Hence, data-dependent methods have become more popular than data-independent methods.

However, most of the existing hashing methods are designed for single-labeled data which cannot be generalized for multi-labeled data directly, such as ImageNet¹, CIFAR-100², IAPRTC-12³, MIRFLICKER⁴, and NUS-WIDE⁵. In fact, most existing hashing methods take two images as similar if they share at least one label as shown in Fig. 1(a). However, it is a natural assumption that two images are more similar if they share more common labels. A multi-labeled example has been shown in Fig. 1, I_a has labels “sky, mountain, tree, river”, I_b has labels “sky, mountain, tree, river”, and I_c has labels “mountain, tree, cloud”. The three images are considered to have great similarity because they share the common labels “mountain, tree”. However, their similarity should be in different degrees according to the number of common labels. Since I_a and I_b has more common labels than I_a and I_c , I_a and I_b are more similar.

Moreover, most hashing methods are based on hand-crafted features such as Gist, Sift and color moments. Those hand-crafted features are costing but not optimal. Recently, deep learning has been widely used for tasks on images. It can be also used to extract the data features.

To solve the above two issues, we proposed a novel supervised hashing method for multi-labeled data. Specifically, we designed a novel feature extraction

¹ <http://www.image-net.org/>.

² <https://www.cs.toronto.edu/~kriz/cifar.html>.

³ <http://imageclef.org/SIAPRdata>.

⁴ <http://press.liacs.nl/mirflickr/mirdownload.html>.

⁵ <http://lms.comp.nus.edu.sg/research/NUS-WIDE.htm>.

algorithm to obtain the “order-preserving” features by using deep convolutional neural network. Extensive experiments on two real-world public datasets show that the proposed method outperforms state-of-the-art baselines in the image retrieval task.

2 Related Work

The existing hashing methods can be grouped into data-independent methods and data-dependent methods. Early works focus on data-independent methods to learn hash functions without using any training data. Representative data-independent methods include LSH [1, 5, 9], shift-invariant kernels hashing (SIKH) [19], and lots of extensions [4, 11, 12, 19]. Different from data-independent methods, data-dependent methods learn hash functions from training data. Using the same length of hash codes or smaller ones, the data-dependent methods can achieve comparable or even better performance comparing to data-independent methods.

Furthermore, existing data-dependent methods can be further divided into three categories: unsupervised methods, supervised methods and semi-supervised methods. Unsupervised hashing tries to preserve the Euclidean similarity between the training points, while supervised hashing [15, 18, 23] and semi-supervised try to preserve the semantic similarity constructed from the semantic labels of the training points.

Unsupervised methods use unlabeled data to learn hash functions and try to keep the neighborhood relations of data in the original space. Representative unsupervised hashing methods include K-means hashing (KMH) [7], Iterative Quantization (ITQ) [6], Spherical Hashing (SH) [8], Discrete Graph Hashing (DGH) [16] and Asymmetric Innerproduct Binary Coding (AIBC) [21], etc. Supervised hashing approaches utilize the label information to build the similarity matrix of training data and further to learn the hash functions. Notable methods in this category include Kernel-Based Supervised Hashing (KSH) [17], Latent Factor Hashing (LFH) [23] and Column Sampling Based Discrete Supervised Hashing (COSDISH) [10], etc. Semi-supervised methods use both labeled and unlabeled data to train hashing functions. Usually, supervised hashing methods achieve higher accuracy than unsupervised methods due to the semantic gap problem. Hence, many recent works focus on supervised methods.

Recently, deep hashing methods have been proposed to perform feature learning and hash codes learning simultaneously. Those methods usually make use of convolutional neural networks and force the output to be hash codes. After iterative training, they can get similarity-preserving hash codes. Deep Pairwise Supervised Hashing (DPSH) [14] and Deep Supervised Ranking Hashing (DSRH) [24] are the representatives of deep hashing.

3 Our Method

3.1 Framework

Figure 2 shows the overall framework of our method. It consists of two part: order-preserving feature learning and hash codes learning.

- **Order-preserving Feature Learning.** As shown in Fig. 2(a), this part contains one input layer, five convolutional layers, two full-connected layers, and one output layer. The raw image will be reshaped to 224×224 as input. The 1st convolutional layer contains 64 filters whose size is 5×5 and the stride is four pixels. We use 256 filters in the 2nd–5th convolutional layers and the stride is one pixel. The max-pooling operator size is set as 2×2 for the 1st, 4th and 5th convolutional layers. Following the CNN part, two full-connected layers contain 4,096 hidden units follow for each. The activation function for all hidden layers is the rectification linear unit (RELU). The last layer is the output whose length is 512. The Gaussian activation function is used for the output layer.
- **Hash Codes Learning.** As shown in Fig. 2(b), the data features generated from the order-preserving feature learning part would become the input of hashing codes learning part. In this part, a supervised hashing algorithm based on maximum likelihood function is designed to optimize hash codes.

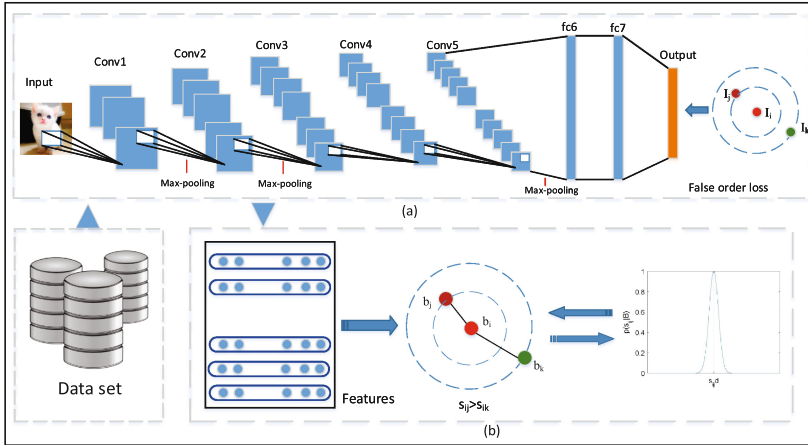


Fig. 2. The framework of our method. Part (a) shows the deep architecture of neural network that produces the order-preserving feature by taking raw images as input. Part (b) show the architecture which generates hash codes for multi-labeled images.

3.2 Similarity Definition

Given a dataset $\{\mathbf{X}, \mathbf{L}\}$, $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ with N points, \mathbf{x}_i is the pixels matrix of the i^{th} image. \mathbf{L} is a set consisting of l_1, l_2, \dots, l_N in which l_i is the label set of the point \mathbf{x}_i .

In the light of multi-labeled data, the intuitional assumption is that the more common labels two labels share, the higher their similarity is. Thus, the similarity between two points can be defined as the proportion of the common labels in the whole labels:

$$s_{ij} = 2 \times \frac{|l_i \cap l_j|}{|l_i \cup l_j|} - 1, \quad (1)$$

where l_i is the labels of the point \mathbf{x}_i . $|l|$ is the size of set l . According to the above equation, $\mathbf{S} = \{s_{ij}\} \in \mathbb{R}^{N \times N}$ can be calculated. $s_{ij} \in [-1, 1]$ represents the degree of similarity between the points \mathbf{x}_i and \mathbf{x}_j . The bigger s_{ij} is, the more similar points \mathbf{x}_i and \mathbf{x}_j are.

3.3 Order-Preserving Feature Learning

For a given data \mathbf{x}_i , the features $\mathbf{u}_i \in \mathbb{R}^{512}$ can be get by passing it to the convolutional neural network. Assume there are three items $\mathbf{x}_i, \mathbf{x}_j$ and \mathbf{x}_k , their output feature are $\mathbf{u}_i, \mathbf{u}_j$ and \mathbf{u}_k respectively. We define the “Order-preserving feature” as below:

Definition 1. For any point \mathbf{x}_i , if there are two other points \mathbf{x}_j and \mathbf{x}_k with the similarities s_{ij} and s_{ik} respectively, such that, their features $\mathbf{u}_i, \mathbf{u}_j$ and \mathbf{u}_k satisfy:

$$(s_{ij} - s_{ik})(E_{ij} - E_{ik}) < 0,$$

where $E_{ij} = \|\mathbf{u}_i - \mathbf{u}_j\|_F^2$, then $\mathbf{u}_i, \mathbf{u}_j$ and \mathbf{u}_k are order-preserving features.

Simply speaking, the “order-preserving” means that if two points have more common labels, they should be close in the feature space than with other points. There are four distance relations between their features:

Table 1. The distances between points in the feature space.

Index	Similarity	Distance	Order-preserving
1	$s_{ij} > s_{ik}$	$E_{ij} > E_{ik}$	False
2	$s_{ij} > s_{ik}$	$E_{ij} < E_{ik}$	True
3	$s_{ij} < s_{ik}$	$E_{ij} > E_{ik}$	True
4	$s_{ij} < s_{ik}$	$E_{ij} < E_{ik}$	False

In Table 1, $s_{ij} > s_{ik}$ represents that point \mathbf{x}_i is more similar with point \mathbf{x}_j than with point \mathbf{x}_k . In this case, \mathbf{u}_i should be closer to \mathbf{u}_j than \mathbf{u}_k as shown in the 2nd and 3rd lines. Considering the two wrong cases in the 1st and 4th lines, the loss function of order-preserving feature can be defined as below:

$$J_{OF} = \frac{1}{N^3} \sum_{i=1}^N \sum_{j=1}^N \sum_{k=1}^N \max((E_{ij} - E_{ik})(s_{ij} - s_{ik}), 0) \quad (2)$$

where $\max(\cdot, \cdot)$ function returns the bigger value.

Algorithm 1. Order-preserving Feature Learning

Require: Dataset $\{\mathbf{X}, \mathbf{L}\}$, minibatch size (128 default), learning rate (0.001 initiated), max iterations (100 default)

Ensure: Data features \mathbf{U}

- 1: calculate similarity \mathbf{S} according to Eqn.(1);
- 2: **repeat**
- 3: Randomly sample a minibatch of points from \mathbf{X} , name as $\mathbf{X}_{\text{batch}}$;
- 4: Calculate features for $\mathbf{X}_{\text{batch}}$ by forward propagation, name as $\mathbf{U}_{\text{batch}}$;
- 5: For \mathbf{u}_i in $\mathbf{U}_{\text{batch}}$:
- 6: Compute derivatives according to Eqn.(3);
- 7: Update the network parameters by utilizing back propagation;
- 8: Endfor
- 9: **until** max iteration number;
- 10: Put all data into network to generate data features.

Optimization. To solve the optimization problems listed above, we employed the stochastic gradient descent (SGD) to minimize the objective function. In Eq. (2), for any items $\mathbf{x}_i, \mathbf{x}_j$ and \mathbf{x}_k , if

$$(E_{ij} - E_{ik})(s_{ij} - s_{ik}) > 0,$$

the derivatives of Eq. (2) with respect to output vectors \mathbf{u}_i are given by:

$$\frac{\partial J_{OF}}{\partial \mathbf{u}_i} = 2(s_{ij} - s_{ik})(\mathbf{u}_k - \mathbf{u}_j) \quad (3)$$

These derivative values can be fed into the underlying CNN via back-propagation algorithm to update the parameters. Algorithm 1 summarizes the process of the order-preserving feature learning part.

3.4 Hashing for Multi-labeled Data

The target of hashing for multi-labeled data is to learn the binary codes $\mathbf{b}_i \in \{-1, 1\}^d$ for each point \mathbf{x}_i , where d is the code length. The inner product between two points \mathbf{b}_i and \mathbf{b}_j can be written as below:

$$\theta_{ij} = \mathbf{b}_i^T \mathbf{b}_j \quad (4)$$

Suppose $\mathbf{B} = \{\mathbf{b}_i\}_{i=1}^N$ is the hash codes for all data, then we can define the likelihood for multi-labeled data as Eq. (5):

$$p(s_{ij}|\mathbf{B}) = e^{-(\theta_{ij} - s_{ij}d)^2}. \quad (5)$$

The likelihood in Eq. (5) forces the θ_{ij} to be an appropriate value and further adjust the hash codes. It significantly differs from the likelihood on singled-labeled data as shown in Fig. 3.

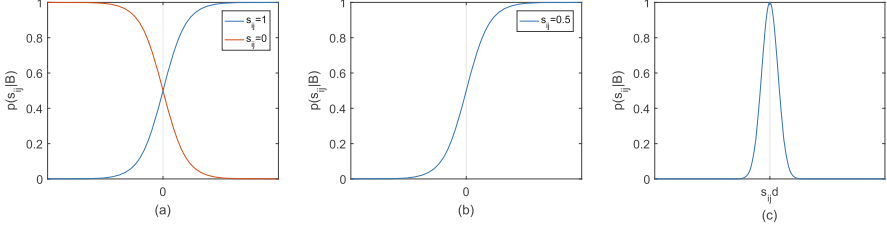


Fig. 3. (a) Traditionally, the likelihood is defined as $p(s_{ij}|\mathbf{B}) = a_{ij}^{s_{ij}}(1 - a_{ij})^{1-s_{ij}}$ s.t. $s_{ij} \in \{0, 1\}$ where $a_{ij} = 1/(1 + e^{-0.5\theta_{ij}})$. (b) For partially similar points, traditional methods treat them as $s_{ij} = 0$ if they share at least one label. This way will weaken the inherent semantic difference and conflict with the characteristics of multi-labeled data. (c) Our likelihood pushes the inner product between two points close to $s_{ij}d$.

By taking \log function on Eq. (5), the objective function of hashing for multi-labeled data would be:

$$\begin{aligned}
 J_{hash} &= \log p(\mathbf{B}|\mathbf{S}) \\
 &= \log p(\mathbf{S}|\mathbf{B})p(\mathbf{B}) \\
 &= - \sum_{s_{ij} \in S} (\theta_{ij} - s_{ij}d)^2 + \log p(\mathbf{B}).
 \end{aligned} \tag{6}$$

Optimization. Since \mathbf{B} is discrete, Eq. (6) is not a convex to optimize. Utilizing the techniques in [23], \mathbf{B} should be relaxed to be \mathbf{H} and $\mathbf{B} = \text{sgn}(\mathbf{H})$. We put a normal distribution on $p(\mathbf{H})$. The Eq. (6) should finally become:

$$J_{hash} = - \sum_{s_{ij} \in S} (\theta_{ij} - s_{ij}d)^2 - \frac{1}{2\lambda} \|\mathbf{H}\|_F^2 + c, \tag{7}$$

where λ is the hyper-parameter. The gradient vector and the Hessian matrix of the objective function J_{hash} with respect to \mathbf{h}_i can be derived as:

$$\begin{aligned}
 \mathbf{D}^1 &= \frac{\partial J}{\partial \mathbf{h}_i^T} = -2 \sum_{j: s_{ij} \in S} (\mathbf{h}_i^T \mathbf{h}_j - s_{ij}d) \mathbf{h}_j^T - 2 \sum_{j: s_{ji} \in S} (\mathbf{h}_j^T \mathbf{h}_i - s_{ji}d) \mathbf{h}_j^T - \frac{1}{\lambda} \mathbf{h}_i^T, \\
 \mathbf{D}^2 &= \frac{\partial^2 J}{\partial \mathbf{h}_i^T \partial \mathbf{h}_i} = -2 \sum_{j: s_{ij} \in S} \mathbf{h}_j \mathbf{h}_j^T - 2 \sum_{j: s_{ji} \in S} \mathbf{h}_j \mathbf{h}_j^T - \frac{1}{\lambda} \mathbf{I}.
 \end{aligned}$$

It can be found that the Hessian matrix is a negative definite matrix. So, this surrogate learning algorithm can be viewed as a generalization of the expectation maximization (EM) algorithm.

- **E-step:** update \mathbf{h}_i with the following rule: $\mathbf{h}_i(t+1) = \mathbf{h}_i(t) - \mathbf{D}^1(t)^T \mathbf{D}^2(t)^{-1}$.
- **M-step:** Compute J_{hash} in Eq. (7) using the new \mathbf{h}_i .

The updating propose can be controlled by the maximum allowed number of iteration T . The initial values of \mathbf{H} can be obtained through PCA on the order-preserving features \mathbf{U} .

4 Experiments

4.1 Datasets and Evaluation Metrics

We evaluate the proposed method on two public datasets of multi-labeled images:

- **CIFAR-100** has 60,000 images in total. We randomly selected 10,000 images as the test query set, and the rest images are used as training samples. Since the labels in CIFAR-100 is hierarchical, we ignored the hierarchy and taken each image as double-labeled.
- **IAPRTC-12** includes the 20,000 segmented images. It is a multi-labeled dataset in which each image has been manually segmented and the resultant regions have been annotated according to a predefined vocabulary of labels. We randomly sampled 2,000 query images as testset and used the rest as the training set. The labels of images in this dataset are hierarchical and multiple. In experiments, we uses the last label of each hierarchy as the label of images. So, each data has 4.1239 labels in average. Since few images share the same label set, the similarity is small but higher than dissimilarity. It makes few positive examples in the training data. To balance the positive and negative examples, the similarity has been strengthened by using $a(s_{ij}) = 1/(1 + e^{-5(1+s_{ij})/2})$.

We measured the performance of methods by Average Cumulative Gain (ACG), Normalized Discounted Cumulative Gain (NDCG) [20], Mean Average Precision (MAP) and Weighted mAP [13].

4.2 Experimental Settings

We implemented the proposed method based on the open-source MatConvnet⁶ framework. The batch size is 128. In training, the weights of the layers are initialized by the pre-trained imagenet-matconvnet-vgg-m⁷ model. The learning rate is initialized to be 0.001. After every 10 epochs on the training data, the learning rate is adjusted to one third of the current value. λ is set as 1.

4.3 Results on Accuracy

In our experiments, some state-of-the-art hashing methods such as DPSH, COS-DISH, LFH, KMH, and ITQ are involved as comparison. The first three methods are supervised and the rest two are unsupervised methods. For all supervised baselines, two images are similar if they share at least one common label, which is set as their authors do. We resize all images to be 224×224 pixels and directly use the raw images as input for DPSH. The rest hashing methods use 512-D gist features as input.

⁶ <http://www.vlfeat.org/matconvnet/>.

⁷ <http://www.vlfeat.org/matconvnet/pretrained/>.

Table 2 shows the comparison results w.r.t. MAP, Weighted mAP, ACG@100, and NDCG@100 on CIFAR-100. As shown in this table, our method achieves the best performance at most cases. But in 32 bits, our method works not good as COSDISH. It is because when the code length is short, the interval information of multi-labeled data cannot be expressed well. The shorter hash codes may lose the information contained in multi-labeled data. With code length grows, the performance of our method increases quickly. Taking NDCG@100 as example, the effectiveness at 64 bits and 128 bits increases by 17.04%, 8.80% respect to the value of 32 bits and 64 bits respectively. Moreover, DPSH works well on average in short code lengths. But our method can achieve comparable results with longer code length as shown in Table 3. Compared with single-labeled data, the information in the multi-labeled data works in the form of the different degrees of similarity. Thus, for multi-labeled data, long hash codes should be generated.

The experimental results on IAPRTC-12 are shown in Table 4. Compared with all baselines, our method takes down the best effectiveness at most criterias. Similar with CIFAR-100, the performance increases quickly with the length of hash codes growing.

Table 2. MAP, Weighted mAP, ACG@100, and NDCG@100 values on CIFAR-100 varying different code lengths.

Method	MAP			Weighted mAP		
	32	64	128	32	64	128
COSDISH	0.2571	0.2327	0.2690	0.3085	0.2792	0.3227
LFH	0.2086	0.2251	0.2508	0.2503	0.2701	0.3010
KMH	0.0700	0.0701	0.0680	0.0881	0.0889	0.0862
ITQ	0.0734	0.0763	0.0789	0.0928	0.0972	0.1010
DPSH	0.4367	0.4504	0.4595	0.5384	0.5593	0.5757
Ours	0.2113	0.4684	0.5840	0.2537	0.5639	0.7007
Method	ACG@100			NDCG@100		
	32	64	128	32	64	128
COSDISH	0.2128	0.1794	0.2257	0.5112	0.4939	0.4888
LFH	0.1513	0.1658	0.1884	0.4721	0.4848	0.4967
KMH	0.1801	0.2017	0.2095	0.4075	0.4097	0.4053
ITQ	0.1900	0.2162	0.2370	0.4110	0.4172	0.4222
DPSH	0.6905	0.7313	0.7714	0.7007	0.7103	0.7171
Ours	0.2152	0.5101	0.6430	0.4663	0.6367	0.7247

4.4 Results of Order-Preserving Features

We further compare the proposed order-preserving feature with hand-craft features (512-D gist feature). We used the order-preserving feature as the input for

Table 3. MAP, Weighted mAP, ACG@100, and NDCG@100 values on CIFAR-100 at 160 bits and 256 bits.

Methods	MAP		Weighted mAP		ACG@100		NDCG@100	
	160	256	160	256	160	256	160	256
DPSH	0.4726	0.5057	0.5935	0.6379	0.7938	0.8452	0.7226	0.7365
Ours	0.6304	0.6709	0.7558	0.8057	0.7190	0.7518	0.7526	0.7762

Table 4. MAP, Weighted MAP, ACG@100, and NDCG@100 values on IAPRTC-12 varying different code lengths.

Method	MAP			Weighted MAP		
	32	64	128	32	64	128
COSDISH	0.4208	0.4549	0.4753	0.5982	0.6546	0.6912
LFH	0.4557	0.4741	0.4793	0.6464	0.6807	0.6914
KMH	0.3350	0.3297	0.3237	0.4629	0.4549	0.4450
ITQ	0.3508	0.3539	0.3559	0.4898	0.4959	0.5005
DPSH	0.5183	0.5268	0.5327	0.7642	0.7777	0.7907
Ours	0.5116	0.5501	0.5659	0.7504	0.8163	0.8555
Method	ACG@100			NDCG@100		
	32	64	128	32	64	128
COSDISH	0.6270	0.7827	0.8361	0.5326	0.5575	0.5676
LFH	0.8188	0.8662	0.8802	0.5527	0.5583	0.5717
KMH	0.5999	0.6000	0.5876	0.4847	0.4792	0.4724
ITQ	0.6281	0.6497	0.6649	0.5020	0.5071	0.5107
DPSH	1.0982	1.1458	1.1789	0.6278	0.6356	0.6404
Ours	1.0940	1.1868	1.2875	0.6048	0.6288	0.6462

baselines. To separate these new versions from existed baselines, we renamed the new versions by adding “+CNN”, such as “COSDISH+CNN”.

Figure 4 illustrates the curves of MAP, Weighted mAP, ACG@100, and NDCG@100 on CIFAR-100. The dotted lines show the results using order-preserving features. The solid lines represent the results using hand-crafted features. It can be found that by using the order-preserving feature, all baselines achieve higher values at most cases. It proved that hand-crafted features cannot capture the inherent characteristics of images.

Figure 5 illustrates the curves of MAP, Weighted mAP, ACG@100, and NDCG@100 on IAPRTC-12. Similar with the curves on CIFAR-100, the order-preserving features bring in better performance. The order-preserving features are more suitable than hand-crafted features on image retrieval tasks.

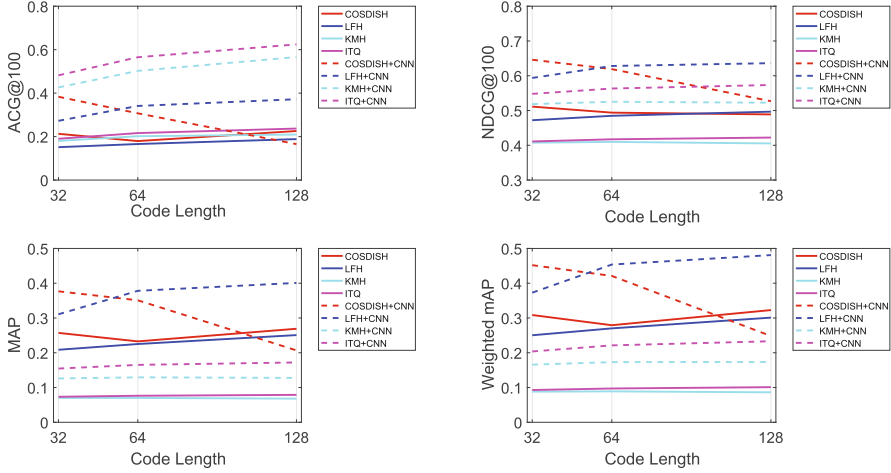


Fig. 4. Curves of MAP, Weighted mAP, ACG@100, and NDCG@100 on CIFAR-100 varying different code lengths after introducing order-preserving features.

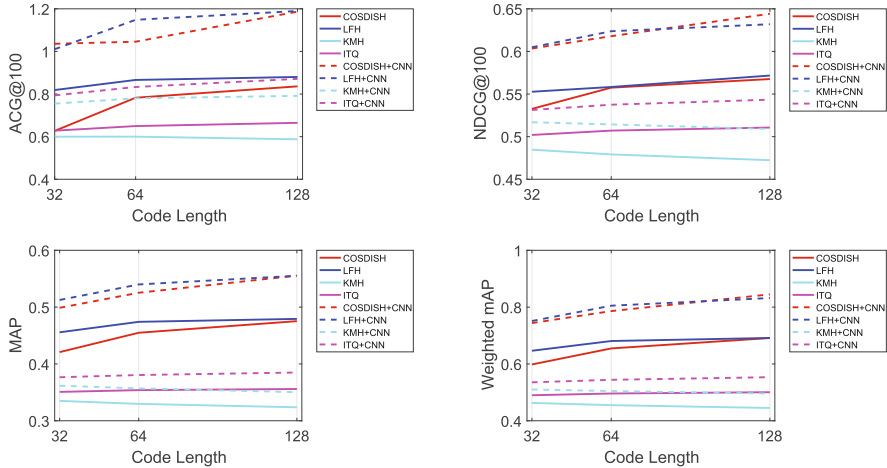


Fig. 5. Curves of MAP, Weighted mAP, ACG@100, and NDCG@100 on IAPRTC-12 varying different code lengths after introducing order-preserving features.

5 Conclusion

In this paper we have proposed a supervised hashing algorithm for multi-labeled images. The CNN model is used to learn order-preserving feature representations. We used some classic ranking metrics to evaluate the performance of our method. The experiment result shows that the characteristics of multi-labeled

data are important to increase the effectiveness of feature representation and hash functions learning.

Acknowledgement. This work was supported by 863 Program (2015AA015404), 973 Program (2013CB329303), China National Science Foundation (61402036, 60973083, 61273363), Beijing Advanced Innovation Center for Imaging Technology (BAICIT-2016007).

References

1. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In: Foundations of Computer Science Annual Symposium, vol. 51, no. 1, pp. 459–468 (2006)
2. Arya, S., Mount, D.M., Netanyahu, N.S., Silverman, R., Wu, A.Y.: An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM (JACM)* **45**(6), 891–923 (1998)
3. Beis, J.S., Lowe, D.G.: Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In: 1997 IEEE Computer Society Conference on Proceedings of Computer Vision and Pattern Recognition, pp. 1000–1006. IEEE (1997)
4. Datar, M., Immorlica, N., Indyk, P., Mirrokni, V.S.: Locality-sensitive hashing scheme based on p-stable distributions. In: Proceedings of the Twentieth Annual Symposium on Computational Geometry, pp. 253–262. ACM (2004)
5. Gionis, A., Indyk, P., Motwani, R.: Similarity search in high dimensions via hashing. In: International Conference on Very Large Data Bases, pp. 518–529 (2000)
6. Gong, Y., Lazebnik, S.: Iterative quantization: a procrustean approach to learning binary codes. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 817–824 (2011)
7. He, K., Wen, F., Sun, J.: K-means hashing: an affinity-preserving quantization method for learning binary compact codes. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 2938–2945 (2013)
8. Heo, J.P., Lee, Y., He, J., Chang, S.F., Yoon, S.E.: Spherical hashing. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2957–2964, June 2012
9. 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. ACM (1998)
10. Kang, W.C., Li, W.J., Zhou, Z.H.: Column sampling based discrete supervised hashing. In: AAAI Conference on Artificial Intelligence (2016)
11. Kulis, B., Grauman, K.: Kernelized locality-sensitive hashing for scalable image search. In: 2009 IEEE 12th International Conference on Computer Vision, pp. 2130–2137. IEEE (2009)
12. Kulis, B., Jain, P., Grauman, K.: Fast similarity search for learned metrics. *IEEE Trans. Pattern Anal. Mach. Intell.* **31**(12), 2143–2157 (2009)
13. Lai, H., Yan, P., Shu, X., Wei, Y., Yan, S.: Instance-aware hashing for multi-label image retrieval. *IEEE Trans. Image Proces.* **25**(6), 2469–2479 (2016). <https://doi.org/10.1109/TIP.2016.2545300>
14. Li, W.J., Wang, S., Kang, W.: Feature learning based deep supervised hashing with pairwise labels. In: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9–15 July 2016, pp. 1711–1717 (2016)

15. Lin, G., Shen, C., Shi, Q., van den Hengel, A., Suter, D.: Fast supervised hashing with decision trees for high-dimensional data. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1971–1978, June 2014
16. Liu, W., Mu, C., Kumar, S., Chang, S.F.: Discrete graph hashing. *Adv. Neural Inf. Process. Syst.* **4**, 3419–3427 (2014)
17. 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 (CVPR), pp. 2074–2081, June 2012
18. Norouzi, M., Blei, D.M.: Minimal loss hashing for compact binary codes. In: Proceedings of the 28th International Conference on Machine Learning (ICML 2011), pp. 353–360 (2011)
19. Raginsky, M., Lazebnik, S.: Locality-sensitive binary codes from shift-invariant kernels. In: Advances in Neural Information Processing Systems 22: Proceedings of a Meeting Held 7–10 December 2009, Vancouver, British Columbia, Canada, pp. 1509–1517 (2009)
20. Rvelin, K., Kekäläinen, J.: IR evaluation methods for retrieving highly relevant documents. In: International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 41–48 (2000)
21. Shen, F., Liu, W., Zhang, S., Yang, Y.: Learning binary codes for maximum inner product search. In: IEEE International Conference on Computer Vision (2015)
22. Wang, J., Yang, J., Yu, K., Lv, F., Huang, T., Gong, Y.: Locality-constrained linear coding for image classification. In: 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3360–3367. IEEE (2010)
23. Zhang, P., Zhang, W., Li, W.J., Guo, M.: Supervised hashing with latent factor models. In: International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 173–182 (2014)
24. Zhao, F., Huang, Y., Wang, L., Tan, T.: Deep semantic ranking based hashing for multi-label image retrieval, pp. 1556–1564 (2015). IEEE Computer Society
25. Zheng, L., Wang, S., Tian, L., He, F., Liu, Z., Tian, Q.: Query-adaptive late fusion for image search and person re-identification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1741–1750 (2015)

Social Media Processing

6th National Conference, SMP 2017, Beijing, China,

September 14-17, 2017, Proceedings

Cheng, X.; Ma, W.-Y.; Liu, H.; Shen, H.-W.; Feng, S.; Xie,
X. (Eds.)

2017, XIII, 356 p. 78 illus., Softcover

ISBN: 978-981-10-6804-1