Image Hashing via Linear Discriminant Learning

Weixiang Hong[*]
National University of Singapore
weixiang.hong@outlook.com

Yu-Ting Chang
UC Merced
ychang39@ucmerced.edu

Haifang Qin[*]
Peking University
qhfpku@pku.edu.cn

Wei-Chih Hung
UC Merced
whung8@ucmerced.edu

Yi-Hsuan Tsai
NEC Labs America
wasidennis@gmail.com

Ming-Hsuan Yang
UC Merced/Google
mhyang@ucmerced.edu

Abstract

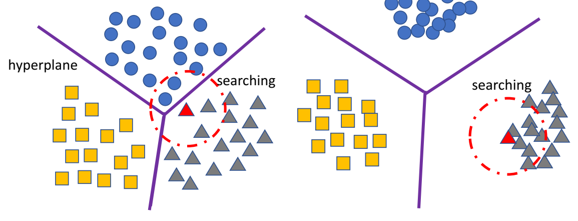
Hashing has attracted attention in recent years due to the rapid growth of image and video data on the web. Benefiting from recent advances in deep learning, deep supervised hashing has achieved promising results for image retrieval. However, existing methods are either less efficient in data usage or incapable of learning linearly discriminative binary codes. In this paper, we revisit linear discriminative analysis and propose a linear discriminative hashing (LDH) objective that is efficient in training and achieves better accuracy in retrieval. With the joint supervision of a classification loss, we design a robust deep network to obtain binary codes that are inter-class separable and intra-class compact, which provides better representations for image retrieval. We conduct extensive experiments on three benchmark datasets, and our LDH algorithm performs favorably against existing state-of-the-art deep supervised hashing methods.

1. Introduction

Image hashing has attracted attention in recent years due to the rapid growth of media data on the web [33, 13, 23, 29, 10, 36, 9, 2]. Generally, hashing aims to encode images/videos into compact binary codes while preserving their mutual similarities. Due to the storage efficiency and low computational cost of compact binary codes, hashing has become one of the most widely-used techniques for image or video search. Existing hashing methods can be grouped into two categories: data-independent and data-dependent approaches. Data-independent methods such as locality sensitive hashing (LSH) [12] and its variants [18, 15, 26] rely on random projections to construct hash functions and do not require training data. In contrast, recent methods focus on the data-dependent scheme that exploits various machine learning techniques to learn more effective hash functions based on a given dataset and usually achieves better performance. These methods can be categorized as supervised and unsupervised schemes. When annotations are not available, unsupervised methods [13, 3] seek various metrics to supervise the learning of hash functions. On the other hand, to utilize the semantic labels of the training data, several supervised hashing methods [24, 29] are proposed.

Following this trend, thanks to the joint learning of feature extraction and hash functions [2, 20, 36, 39], deep supervised hashing has demonstrated state-of-the-art retrieval performance over conventional methods. Since the "ground-truth" hash codes are not available, existing methods usually consider hash codes as representations to an auxiliary task such as pair-wise/triplet similarity estimation [2, 39, 32, 38] and image classification [20, 36, 37]. However, such auxiliary tasks may introduce drawbacks, such as the loss in training efficiency or accuracy.

On one hand, the pair-wise/triplet similarity estimation methods [2, 39, 32, 38] require heavy computation of inefficient data sampling, e.g., the pair-wise hashing methods



(a) Classification-based methods (b) Our LDH method

Figure 1. **Illustration of our motivation.** Compared with existing classification-based hashing methods, the projected data by the proposed LDH method is intra-class compact and inter-class separable. Thus, given a query (denoted as the red rectangle), even with the same classification accuracy, our method achieves better retrieval performance.

[*]Work carried out at Google.

require $\mathrm{C}_n^2 = \frac{n(n-1)}{2}$ sample pairs to train a model for n images, *i.e.*, with the data complexity of $\mathcal{O}(n^2)$. The data complexity becomes even more significant with $\mathcal{O}(n^3)$ for the triplet hashing methods. In addition, it takes an enormous amount of time to sample enough pairs or triplets for training. Without sufficient samples, hashing methods tend to fit the mutual similarities locally and fail to model global affinities, thereby degrading the retrieval performance. On the other hand, the classification-based methods [20, 36, 37] take binary codes as image representations for discrimination, which is more efficient in data utilization. Nevertheless, it is not clear whether learning such hash codes via the classification task is optimal for image/video retrieval. As shown in Figure 1(a), hash codes that are perfectly separated may not work well for the retrieval application.

We notice that rare efforts have been made to learning discriminative binary codes in recent classification-based hashing methods. In this work, we analyze the reasons for the retrieval performance loss of classification-based methods and thus propose a linear discriminative hashing objective. Our intuition is to incorporate the linear discriminative analysis on top of a deep convolutional neural network (CNN), which aims to produce linearly discriminative binary codes as shown in Figure 1(b). In addition to maximizing the likelihood of target labels for individual samples, we minimize intra-class variances and maximize inter-class variances *w.r.t.* centers of hash codes. We update the centers of hash codes by accumulating gradients in stored decimal centers to maintain the precision, meanwhile training the CNN with binary centers during forward and backward passes. By encouraging the network to generate discriminative hash codes, our LDH method performs favorably against state-of-the-art deep hashing methods.

The contributions of this work are as follows. First, we propose the linear discriminative hashing algorithm to learn hash codes that are intra-class compact and inter-class separable. The proposed algorithm is efficient in data usage and accurate in retrieval performance. Second, we introduce a runtime sampling approach to effectively update the centers of hash codes, which leads to performance gains. Third, we conduct extensive experiments on three benchmark datasets, including CIFAR-10 [16], ImageNet [28] and NUS-WIDE [4]. Experimental results demonstrate the efficacy of the proposed algorithm.

2. Related Work

2.1. Hashing

Existing hashing methods can be broadly categorized into unsupervised [13, 3] and supervised [24, 29] approaches. The optimization techniques of both categories are similar to some extent, and the key difference is whether data annotations are leveraged to supervise the learning process of hash codes. For example, the Iterative Quantization (ITQ) [13] method rotates the decimal feature vectors to align them with the corresponding binary vertices, while the Binary Autoencoder (BA) [3] algorithm aims to minimize the reconstruction loss by treating hash codes as latent representations in autoencoder. The Supervised Hashing with Kernels (KSH) [24] method learns hash codes by minimizing/maximizing Hamming distances between similar/dissimilar pairs.

Deep hashing, especially deep supervised hashing, has recently demonstrated promising retrieval performance in several benchmarks. A comprehensive review of hashing can be found in [31] and here, we review two representative classification-based deep hashing method that is closely related to our work, namely, Supervised Semantics-preserving Deep Hashing (SSDH) [36] and Deep Discrete Supervised Hashing (DSDH) [20]. The SSDH [36] algorithm constructs hash functions as a latent layer in a deep network and softly binzarizes the output of the latent layer via the sigmoid activation. In addition to minimizing the classification error, SSDH also imposes other desirable properties such as bit independence and bit balance on the hash codes. DSDH [20] takes image classification as surrogate problem for learning binary code. It assumes that good binary code for classification is also effective in retrieval. Different from these work, our LDH method learns hash codes that are intra-class compact and inter-class separable via a linear discriminative objective.

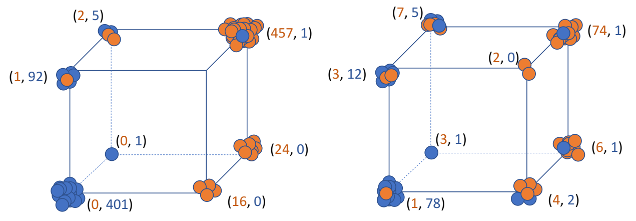
2.2. Linear Discriminative Analysis

Linear Discriminant Analysis (LDA) is developed from multivariate statistics which seeks a linear projection of high-dimensional data samples into a lower-dimensional space [11]. In order to achieve desirable linear decision boundaries, minimum intra-class variance and maximum inter-class variance are introduced to achieve the resulting latent space. In addition, a deep learning-based extension of LDA has been recently proposed in [8]. We also note that an earlier work, LDAHash [30], aims to combine LDA and hashing. However, this method relies on the linear projection upon hand-crafted features and cannot be jointly trained with CNN, which limits its capacity in learning effective hash functions.

To enhance the discriminative strength of deep neural networks, Wen *et al.* [34] propose the center loss to simultaneously learn class centers and penalize the distances of samples to their corresponding class centers. By jointly training the center loss with the classification softmax function, this method achieves state-of-the-art performance in face recognition and verification tasks. A more detailed comparison to our method would be provided in the later section.

Table 1. **Architecture of the light-weight network.** $(11 \times 11)_{3\times32}$ denotes a convolution layer with 32 filters of size 11×11. (2×2) denotes the max-pooling layers with grid of 2×2 and (64×3) stands for a 64×3 fully-connected layer. We use the Rectified Linear Unit (ReLU) as the nonlinear activation function.

Layer type	Conv + ReLU	MaxPool	Conv + ReLU	MaxPool	Fc + Sigmoid	Softmax
Layer shape	$(11 \times 11)_{3\times32}$	(2×2)	$(5 \times 5)_{32\times16}$	(2×2)	(64×3)	(3×2)



(a) Training set distributions. (b) Testing set distributions.

Figure 2. **Distribution of deeply learned hash codes.** (a) The distributions of binary codes in the training set. (b) The distributions of binary codes in the testing set. The circles with different colors denote samples from different classes. The pair of numbers in each bracket represents the number of circles in this vertex (best viewed in color).

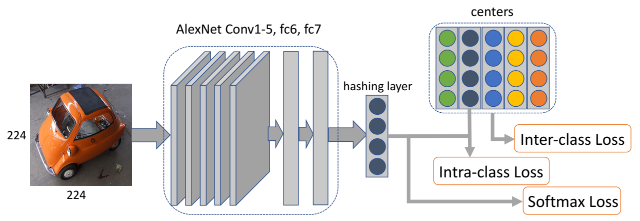


Figure 3. **Linear discriminative hashing.** We propose two loss terms: the intra-class loss and the inter-class loss. The intra-class loss is jointly trained with the conventional softmax loss, while the inter-class loss is applied on the class centers of the binary codes. The final hash codes are obtained via a sigmoid layer to softly binarize the outputs.

3. Proposed Algorithm

In this section, we first illustrate the unsatisfactory distribution of a general deep binary hashing through a toy example. To tackle the issue, we propose the Linear Discriminative Hashing (LDH) algorithm to improve the discriminative power of the learned hash codes. We also introduce an optimization method for updating centers of hash codes. Finally, we discuss the merits of the proposed algorithm.

3.1. Distribution of Deeply Learned Hash Codes

To demonstrate how the distribution of deeply hash codes could be unsatisfactory for retrieval, we conduct an experiment on a toy dataset with the dog and airplane classes sampled from the CIFAR-10 dataset [16], where 600 images are selected for both class with 500 training and 100 testing examples. We build a light-weight network (Table 1) that consists of only two convolutional layers and one fully-connected layer with output channel as 3, followed by a sigmoid layer to softly binarize the outputs into hash codes. The layers configuration are referenced from the AlexNet [17]. We set the length of the binary codes as 3 for visualizing binary codes on a 3-D cube. We train the network with the softmax loss function:

$$L_{cls} = -\sum_{i=1}^{n} \log \frac{e^{W_{y_i}^T x_i + t_{y_i}}}{\sum_{j=1}^{C} e^{W_{y_j}^T x_i + t_{y_j}}}, \qquad (1)$$

where n and C denote the size of mini-batch and the number of class respectively. We denote b as the dimension of binary codes, $x_i \in [0,1]^b$ as the binary codes of i-th sample that belongs to the y_i-th class. In addition, $W_j \in \mathbb{R}^b$ is the j-th column of the weight matrix $W \in \mathbb{R}^{b\times C}$ in the last fully connected layer and $t \in \mathbb{R}^C$ is the bias term.

We present the resulting 3-dimensional binary codes of each class and visualize them in Figure 2. We observe that: 1) under the supervision of softmax loss, the deeply learned binary codes are separable, *e.g.*, 499 out of 500 blue points in the training set can be correctly classified; 2) the binary codes are not sufficiently discriminative, since they still show significant intra-class variations. For example, only 401 out of 499 blue points share the same binary codes, while there are 98 blue points that are correctly classified but not compactly hashed. Consequently, it is not effective to directly use these binary codes for retrieval.

3.2. Linear Discriminative Hashing

As shown in Section 3.1, the ensuing question is to make the learned binary codes more discriminative rather than just linearly separable. Intuitively, it is desirable to group the binary codes of the same class close together, and scatter the binary codes of the different classes far apart. To this end, we propose the linear discriminative hashing objective with the following loss function:

$$L_D = \alpha \sum_{i=1}^{N} ||x_i - c_{y_i}||_2^2 - \beta \sum_{i=1}^{C} \sum_{j=1, j\neq i}^{C} ||c_i - c_j||_2^2, \qquad (2)$$

where N is the number of samples in the training set, and $\{c_i\}_{i=1,\dots,C}$ are a set of binary centers corresponding to each class. The first term in (2) is referred as the intra-class loss since it reflects the variations within each class, while the second term is the inter-class loss as it indicates the inter-class separability. We use α and β to balance the weight of two loss terms. Our final objective is to optimize (1) and (2) with a deep neural network. The binary features are jointly supervised by (1) and intra-class loss, while the centers are optimized *w.r.t.* intra-class loss and inter-class loss.

Since it is inefficient to take the entire training set into account in one iteration, we thus compute the intra-class loss over a mini-batch of data. This modification allows us to update network parameters with mini-batch using stochastic gradient descent (SGD). In Figure 3, we show the overall framework integrating (2) on top of a deep neural network for learning discriminative hash codes.

Nevertheless, it is unclear how to update the centers of hash codes. Ideally, we should update $\{c_{y_i}\}$ during the optimization step. However, since $\{c_{y_i}\}$ are binary vectors in our case, it is not possible to perform SGD on binary centers. In the supplementary material, we show that naively relaxing the binary constraints on centers of hash codes leads to performance drop. To this end, we propose a runtime sampling approach that tackles this problem introduced below.

3.3. Runtime Sampling Approach

The class centers of hash codes are desired to be binary, such that CNN features are encouraged to be discrete. On the other hand, the centers of hash codes should be decimal, such that we can perform SGD optimization. To handle such conflicting factors, we propose to treat the centers of hash codes as binary during forward and backward passes, while using decimal ones during the parameter update.

Specifically, the binary centers are sampled from Bernoulli distributions that use decimal numbers of centers as probabilities during each batch. Similar to the setting of [27, 5], we consider the back-propagation at 3 different steps: 1) forward pass to compute the loss, 2) backward pass to compute the gradients, and 3) parameter update by using gradients of hash code centers. The parameter update at step 3 is performed on the decimal centers, and the sampling of binary centers in the next training batch will be affected accordingly. Although the changes to decimal centers are small via gradient descent, we expect that such updates are accumulated to move the centers of hash codes toward the direction that mostly improves the training objective. Since we need to keep the decimal centers within $[0, 1]$ to be valid probabilities for Bernoulli distributions, we clip the decimal centers within the $[0, 1]$ interval after the update. The decimal centers would otherwise grow too large due to the existence of the inter-class loss. The main steps of our approach are summarized in Algorithm 1.

Algorithm 1 Updating centers of hash codes with SGD

```
Require: a minibatch of (inputs, targets), decimal centers $\mathrm{C}_d$ in previous batch, and learning rate $\eta$
Ensure: decimal centers $\mathrm{C}_d$ is updated
1: For each epoch:
2:   Initialize Bernoulli distributions as
     $\mathrm{m} = \mathrm{Bernoulli}(\mathrm{C}_d)$
3:   For each batch:
4:     Sample binary centers $\mathrm{C}_b = \mathrm{m.sample}()$
5:     Forward with binary centers $\mathrm{C}_b$ and compute the loss $\mathrm{L}$
6:     Backward w.r.t. binary centers $\mathrm{C}_b$ for the gradient $\frac{\partial \mathrm{L}}{\partial \mathrm{C}_b}$
7:     Update $\mathrm{C}_d = \mathrm{C}_d - \eta \frac{\partial \mathrm{L}}{\partial \mathrm{C}_b}$
8:   Clip $\mathrm{C}_d$ to ensure it within $[0, 1]$
9: return
```

3.4. Discussions

Comparison with Center Loss. Our linear discriminative hashing algorithm is different from the method using the center loss [34] in two key aspects. First, the center loss is proposed in the decimal space, and thus it only emphasizes to decrease the intra-class variance and ignores to increase the inter-class variance due to its decimal nature. Second, since each bit in hash codes is supposed to fire 50% of the time [33, 37], it is important to maximize the inter-class variance as it helps fully utilize the discriminative strength of all hashing bits. In [34], the inter-class variance is not well exploited based on the center loss.

Comparison with Triplet Loss and Contrastive Loss. Recently, contrastive loss [6] and triplet loss [32, 38] have been proposed to train deep hashing networks. However, both these losses suffer from significant data expansion when drawing pair or triplet samples from the training set. The proposed linear discriminative hashing method inherits the advantage as the softmax loss, while requiring less complex arrangements of training samples. Consequently, the supervised learning process of our discriminative hashing method is efficient and is easy to implement. Furthermore, our loss function targets more directly at learning compact intra-class and separable inter-class variances, which are effective for learning discriminative hash codes for the retrieval application.

4. Experiments

We conduct extensive experiments to evaluate our algorithm against several state-of-the-art hashing methods on three standard benchmarks. All the source code and trained models will be made available to the public.

4.1. Setup

We evaluate the proposed algorithm on three image retrieval benchmarks: CIFAR-10 [16], ImageNet [28], and NUS-WIDE [4]. We also present ablation studies to ana-

Table 2. Retrieval performance in terms of mean average precision (mAP%) for different code length as 16, 32, 48 and 64 bits.

Method	CIFAR-10				ImageNet				NUS-WIDE			
	16 bits	32 bits	48 bits	64 bits	16 bits	32 bits	48 bits	64 bits	16 bits	32 bits	48 bits	64 bits
SH [33]	12.2	13.5	12.1	12.6	47.9	49.1	49.8	51.5	20.8	32.7	39.5	41.0
CCA-ITQ [13]	31.4	36.1	36.6	37.9	26.6	43.6	54.8	58.0	50.9	54.4	56.8	67.6
DHN [40]	56.8	60.3	62.1	63.5	31.1	47.2	54.2	57.3	63.7	66.4	66.9	67.1
DNNH [19]	55.5	55.8	58.1	62.3	29.7	46.3	54.0	56.6	68.1	71.3	71.8	72.0
DPSH [21]	64.6	66.1	67.7	68.6	32.6	54.6	61.7	65.4	71.5	72.6	73.8	75.3
DSH [22]	68.9	69.1	70.3	71.6	34.8	55.0	62.9	66.5	71.8	72.3	74.2	75.6
HashNet [2]	70.3	71.1	71.6	73.9	50.6	62.9	66.3	68.4	73.3	75.2	76.2	77.6
PGDH [39]	73.6	74.1	74.7	76.2	51.8	65.3	**70.7**	**71.6**	76.1	78.0	78.6	79.2
Ours	**77.5**	**78.4**	**78.8**	**79.2**	**61.8**	**66.7**	69.0	69.0	**76.9**	**78.9**	**78.7**	**80.3**

lyze the contributions of the main modules in the proposed algorithm.

CIFAR-10. The CIFAR-10 [16] dataset contains 60,000 images of 32 $\times$ 32 pixels in 10 categories. Similar to the protocol in Deep Quantization Network [1], we randomly select 100 images per class as the query set, 500 images per class as the training set, and remaining images as the database for retrieval.

ImageNet. The ImageNet [28] dataset is widely used for visual recognition tasks such as the Large Scale Visual Recognition Challenge (ILSVRC 2015). It contains over 1.2M images in the training set and 50K images in the validation set, where each image is single-labeled by one of the 1,000 categories. We randomly select 100 categories, use all the images of these categories in the training set as the database for retrieval and utilize all the images in the validation set as queries. Furthermore, we randomly select 100 images per category from the database as the training set.

NUS-WIDE. The NUS-WIDE [4] dataset contains 269,648 images collected from Flickr. This is a multi-label dataset where each image is associated with one or multiple labels from the given 81 concepts. Similar to the setting in [2, 39], we use a subset of 195,834 images that are associated with the 21 most frequent concepts, where each concept consists of at least 5,000 images. We randomly sample 2,100 images with 100 images per category to form the query set and use remaining images as the database. We uniformly sample 500 images per category from the database to form the training set. Since our method requires to associate each input image to a center, we split an image with multiple labels into multiple training samples, each of which corresponds to one of its label. For images with relatively fewer labels, we draw samples multiple times to keep data balanced. We also discuss alternative strategies for handling multi-label datasets in the later section.

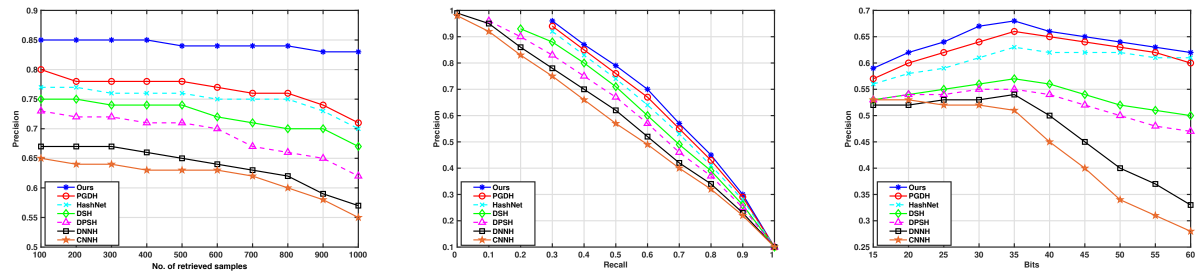
Evaluation Metric. Using the standard evaluation protocol in the literature [35, 19, 1], two images are considered similar if they share at least one semantic label. We evaluate the retrieval performance of generated binary codes using the following metrics: mean average precision (mAP), precision-recall (P-R) curve, precision at top retrieved samples (P@N) and Hamming lookup precision within a Hamming radius $r = 2$ (HLP@2). We evaluate the performance using binary codes of 16, 32, 48 and 64 bits. Note that for the ImageNet dataset, we employ mAP@1000 as each category contains only 1,300 images. For the CIFAR-10 and NUS-WIDE datasets, we adopt mAP@54000 and mAP@5000 respectively.

Implementation Details. For the proposed algorithm, we utilize the AlexNet architecture and implement it using the PyTorch framework. We initialize convolutional layers *conv1 - conv5* and fully-connected layers *fc6 - fc7* with the pre-trained model on ImageNet. The final hashing layer is initialized with the Gaussian distribution. The parameters are obtained by 10-fold cross validation. For the weights in the loss function, we fix $\alpha = 0.01$ and $\beta = 0.001$ for all the experiments and more analysis is provided in the supplementary material. During training, the learning rate for updating CNN and centers of hash codes are set to 0.01 and 0.001, respectively.

4.2. Results and Analysis

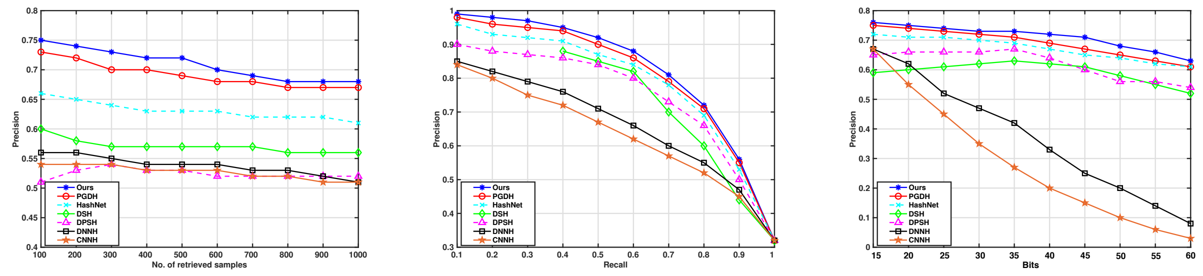
Comparisons with State-of-the-art Methods. We evaluate the proposed LDH algorithm against 8 state-of-the-art hashing methods, including SH [33], CCA-ITQ [13], DHN [40], DNNH [19], DPSH [21], DSH [22], HashNet [2] and PGDH [39]. For DNNH [19] and PGDH [39], we use the reported results in their paper. For all the other methods, we use the released source codes for performance evaluation. For conventional hashing methods like SH [33] and CCA-ITQ [13], we use the DeCAF7 [7] features as input. For deep hashing methods, we directly use raw images as input and resize images to fit the adopted network. Note that we adopt the AlexNet model for all deep hashing methods for fair comparisons unless specified otherwise.

Table 2 shows the retrieval performance of different hashing methods in terms of mAP using different code lengths. Overall, the proposed LDH algorithm performs



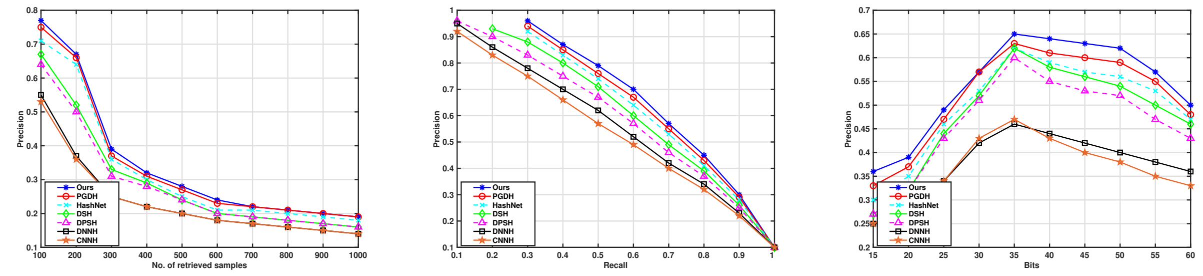
(a) Precision at top retrieved samples (P@N) (b) Precision-recall curve @ 64 bits (c) Precision within Hamming radius 2

Figure 4. Experimental comparisons on the CIFAR-10 dataset using three evaluation metrics.



(a) Precision at top retrieved samples (P@N) (b) Precision-recall curve @ 64 bits (c) Precision within Hamming radius 2

Figure 5. Experimental comparisons on the NUS-WIDE dataset using three evaluation metrics.



(a) Precision at top retrieved samples (P@N) (b) Precision-recall curve @ 64 bits (c) Precision within Hamming radius 2

Figure 6. Experimental comparisons on the ImageNet dataset using three evaluation metrics.

favorably against all evaluated methods. Compared to the state-of-the-art PGDH [39] method, our algorithm performs favorably on the CIFAR-10 and NUS-WIDE datasets, and achieves competitive results on the ImageNet dataset. The performance gains of the proposed LDH are more especially when the hash codes are compact, *e.g*., our mAP is 10% higher than that of PGDH [39] with 16-bit codes on the ImageNet dataset. In addition, deep hashing methods significantly outperform conventional hashing schemes on all three datasets by large margins, even when CNN features are used by all evaluated approaches. The results suggest that the end-to-end learning scheme facilitates in learning effective hash codes for image retrieval.

More Results. The average precision in terms of different numbers of top retrieved results (P@N) is shown in Figure 4(a), 5(a) and 6(a), where the code length is fixed at 64 bits. For presentation clarity, only the results by deep learning based methods are presented. For performance evaluation on all the three datasets, N ranges from 100 to 1,000 here. Overall, the proposed LDH algorithm consistently performs well against all the evaluated hashing methods for the same amount of retrieved samples. We note that more images that have similar semantic labels are retrieved by the proposed LDH algorithm, which is desirable for numerous applications using hash codes.

The evaluation results on the CIFAR-10, NUS-WIDE

Table 3. Effect of different loss functions on the CIFAR-10 dataset.

	16 bits	32 bits	48 bits	64 bits
Cls Loss Only	75.2	75.5	76.0	76.6
Cls Loss + Intra-class Loss	75.1	76.5	76.9	77.8
Intra-class Loss + Inter-class Loss	74.5	75.8	76.2	77.0
Ours (Cls Loss + Inter-class Loss + Intra-class Loss)	**77.5**	**78.4**	**78.8**	**79.2**

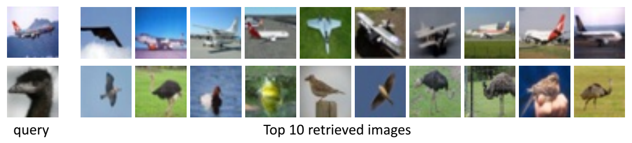


Figure 7. Retrieval results on CIFAR-10 dataset.

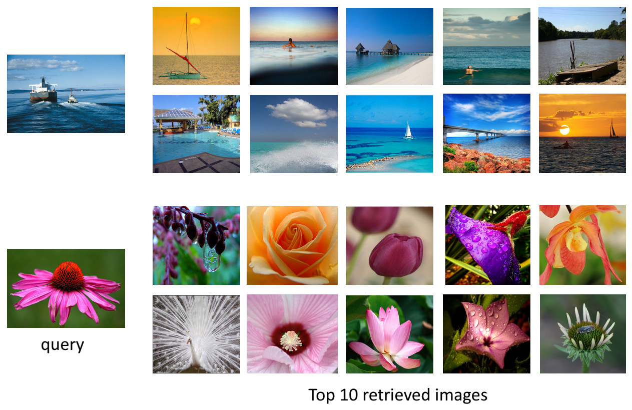


Figure 8. Retrieval results on NUS-WIDE dataset.

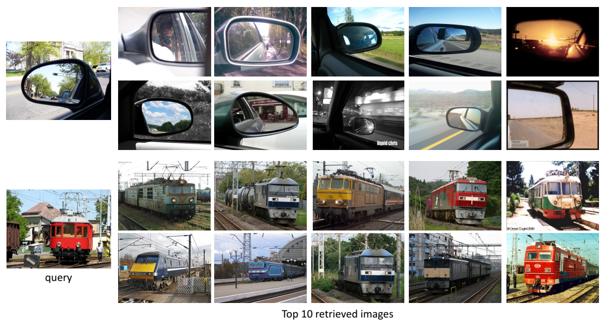
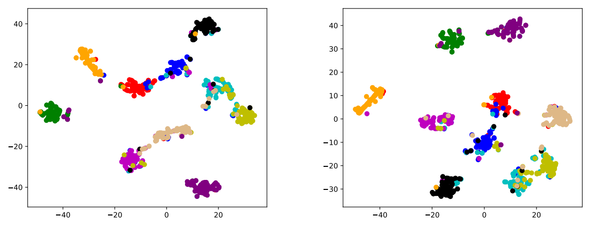


Figure 9. Retrieval results on ImageNet dataset.



(a) HashNet [2]

(b) LDH

Figure 10. **Visualization of hash codes usisng t-SNE.** Our LDH produces more discriminative embeddings that are intra-class compact and inter-class separable. The two classes relatively close in the bottom-right of (b) are automobile and truck.

and ImageNet datasets in terms of Precision-Recall (PR) curves for 64-bit binary codes are shown in Figure 4(b), 5(b) and 6(b). In this setting, LDH performs favorably against all evaluated methods, which shows that effective hash codes are learned for image retrieval.

Figure 4(c), 5(c) and 6(c) show the Hamming lookup precision (within Hamming radius 2, HLP@2) with various code lengths. This evaluation metric measures the precision of retrieved results falling into the buckets within the Hamming radius of 2. The results validate the compactness of binary codes learned by our LDH algorithm. We also observe that the best performance is achieved at a moderate length of binary codes. This is because that longer binary code makes the data distribution in the Hamming space sparse, such that fewer samples fall within the Hamming radius.

Some retrieval examples are shown in Figure 7, 8 and 9. The LDH algorithm is able to retrieve images that share the same semantic labels with the input query. More results are shown in the supplementary material.

Visualization of Hash Codes. We visualize the hash codes generated by the HashNet [2] and LDH methods on the CIFAR-10 dataset using t-SNE [25] in Figure 10. The hash codes generated by our LDH algorithm show clear discriminative embeddings, in which different categories are well separated. The results suggest that LDH learns more discriminative hash codes than HashNet for more accurate image retrieval.

4.3. Ablation Study

We analyze the contributions of different components of the LDH algorithm. In addition, we use the 50-layer Deep residual Net (ResNet-50) [14] as our backbone model for performance evaluation.

4.3.1 Effect of Loss Functions

We experimentally evaluate the contributions of different components of our loss. Table 3 shows the retrieval performance using variants of the LDH algorithm, where we use the same learning rate and batch size for experiments.

As shown in the table, the LDH algorithm consistently

Table 4. Different approaches for applying the LDH algorithm to the multi-label datasets.

	16 bits	32 bits	48 bits	64 bits
BCE Loss (Nearest)	73.9	75.2	76.4	77.3
BCE Loss (Average)	72.4	73.5	73.3	74.5
Softmax	74.2	77.5	77.8	78.6
Balanced Softmax	**76.9**	**78.9**	**78.7**	**80.3**

outperforms the other 3 variants, which shows that all components are necessary for achieving the best retrieval results. Note that the evaluation setting of *Cls Loss* with *Inter-class Loss* is omitted as it is equivalent to *Cls Loss Only*.

4.3.2 Handling Multi-label Dataset

We discuss several alternative strategies for training the proposed LDH algorithm on multi-label dataset such as NUS-WIDE. A straightforward approach is to consider it as a multi-label classification problem and minimize the binary cross entropy (BCE) loss. We propose two methods to compute intra-class loss in the context of BCE Loss, *i.e.*, BCE Loss (Nearest) and BCE Loss (Average). Specifically, for BCE Loss (Nearest), we dynamically select the nearest center according to the hash code of the input image, and then use the nearest center for computing intra-class loss. For BCE Loss (Average), we compute the mean of the intra-class loss to all centers that the input image is labeled to. However, neither one performs well as shown in Table 4.

As the LDH algorithm performs well on the single-label dataset, we split a multi-label image into several single-label ones such that we can train our method using the softmax loss. In Table 4, the *Softmax* approach denotes that each annotation of an image is repeated once, *i.e.*, if a training sample is labeled to 5 semantic classes, it is split into 5 different training entries with distinct labels. Although this Softmax approach already performs significantly better than the ones using the BCE Loss, it tends to ignore images with relatively fewer labels. To address this issue, for each training sample with less than 5 labels, we repeatedly sample it for 5 times and each time we randomly select a label. We call this sampling strategy as *Balanced Softmax*. The improvements in Table 4 demonstrate the effectiveness of this Balanced Softmax approach for using the LDH algorithm on the multi-label dataset.

4.3.3 Effect of Runtime Sampling

We carry out experiments to verify the effectiveness of our runtime sampling approach. Recall that our approach in Algorithm 1 decomposes a propagation process into forward pass, backward pass and parameter update. We adopt binary centers fixed in the forward pass and backward pass, and utilize decimal centers in parameters update. A straightforward relaxation is to adopt decimal centers in all 3 steps, *i.e.*, there are no binary centers involved. This relaxation approach degrades to the method using the center loss [34] if we further remove the inter-class loss. Table 5 shows the results using different approaches.

Our runtime sampling strategy requires the least relaxation on binary constraints on centers, hence achieving the best performance. Also, it can be observed in Table 5 that inter-class loss can help boost the retrieval results, which is consistent with what we observe in Table 3.

Table 5. Different strategies for updating centers of hash codes.

		16 bits	32 bits	48 bits	64 bits
CIFAR-10	center loss	74.2	75.5	76.3	77.0
	center loss + inter-class loss	75.6	77.1	77.9	78.5
	LDH	**77.5**	**78.4**	**78.8**	**79.2**
ImageNet	center loss	53.1	62.0	64.9	65.1
	center loss + inter-class loss	55.6	63.3	66.8	67.8
	LDH	**61.8**	**66.7**	**69.0**	**69.0**
NUS-WDIE	center loss	70.8	75.9	76.5	78.0
	center loss + inter-class loss	73.2	76.4	77.2	78.4
	LDH	**76.9**	**78.9**	**78.7**	**80.3**

Table 6. LDH with different backbone models.

		16 bits	32 bits	48 bits	64 bits
CIFAR-10	AlexNet	77.5	78.4	78.8	79.2
	ResNet-50	**86.9**	**87.2**	**88.3**	**88.1**
ImageNet	AlexNet	61.8	66.7	69.0	69.0
	ResNet-50	**84.9**	**87.0**	**88.1**	**88.0**
NUS-WIDE	AlexNet	74.6	78.3	78.8	79.8
	ResNet-50	**80.0**	**82.6**	**82.3**	**82.4**

4.3.4 ResNet-50 as Backbone Model

It is worth mentioning that most existing state-of-the-art deep hashing methods use the AlexNet [17] as backbone model. We evaluate the proposed LDH algorithm using the ResNet-50 [14] model and present experimental results in Table 6. The ResNet-50 model is pre-trained on the ImageNet. As expected, the proposed algorithm performs better when a stronger backbone model is utilized.

5. Conclusions

In this work, we propose the Linear Discriminative Hashing (LDH) algorithm for image retrieval. By enforcing the intra-class compactness and inter-class separability, the proposed LDH learns linearly discriminative binary codes that are effective for retrieval tasks. A runtime sampling approach is developed to enable the training of binary centers with stochastic gradient descent. We evaluate the LDH algorithm on three benchmark datasets, including CIFAR-10, ImageNet, and NUS-WIDE, with ablation studies to analyze the contributions of main components of our loss functions. Experimental results show that the proposed LDH algorithm performs favorably against other state-of-the-art hashing approaches.

References

[1] Y. Cao, M. Long, J. Wang, H. Zhu, and Q. Wen. Deep quantization network for efficient image retrieval. In *AAAI*, 2016. 5

[2] Z. Cao, M. Long, J. Wang, and P. S. Yu. Hashnet: Deep learning to hash by continuation. In *ICCV*, 2017. 1, 5, 7

[3] M. A. Carreira-Perpinan and R. Raziperchikolaei. Hashing with binary autoencoders. In *CVPR*, 2015. 1, 2

[4] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng. Nus-wide: a real-world web image database from national university of singapore. In *ACM CIVR*, 2009. 2, 4, 5

[5] M. Courbariaux, Y. Bengio, and J.-P. David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *NIPS*, 2015. 4

[6] Q. Dai, J. Li, J. Wang, and Y.-G. Jiang. Binary optimized hashing. In *ACM Multimedia Conference*, 2016. 4

[7] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*, 2014. 5

[8] M. Dorfer, R. Kelz, and G. Widmer. Deep linear discriminant analysis. In *ICLR*, 2016. 2

[9] Y. Duan, J. Lu, Z. Wang, J. Feng, and J. Zhou. Learning deep binary descriptor with multi-quantization. In *CVPR*, 2017. 1

[10] V. Erin Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou. Deep hashing for compact binary codes learning. In *CVPR*, 2015. 1

[11] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936. 2

[12] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *VLDB*, 1999. 1

[13] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*, 2011. 1, 2, 5

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 7, 8

[15] J. Ji, J. Li, S. Yan, B. Zhang, and Q. Tian. Super-bit locality-sensitive hashing. In *NIPS*, 2012. 1

[16] A. Krizhevsky. Learning multiple layers of features from tiny images. *Masters thesis, Department of Computer Science, University of Toronto*, 2009. 2, 3, 4, 5

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 3, 8

[18] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. In *ICCV*, 2009. 1

[19] H. Lai, Y. Pan, Y. Liu, and S. Yan. Simultaneous feature learning and hash coding with deep neural networks. In *CVPR*, 2015. 5

[20] Q. Li, Z. Sun, R. He, and T. Tan. Deep supervised discrete hashing. In *NIPS*, 2017. 1, 2

[21] W.-J. Li, S. Wang, and W.-C. Kang. Feature learning based deep supervised hashing with pairwise labels. In *IJCAI*, 2016. 5

[22] H. Liu, R. Wang, S. Shan, and X. Chen. Deep supervised hashing for fast image retrieval. In *CVPR*, 2016. 5

[23] W. Liu, C. Mu, S. Kumar, and S.-F. Chang. Discrete graph hashing. In *NIPS*, 2014. 1

[24] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *CVPR*, 2012. 1, 2

[25] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *JMLR*, 9:2579–2605, 2008. 7

[26] Y. Mu and S. Yan. Non-metric locality-sensitive hashing. In *AAAI*, 2010. 1

[27] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV*, 2016. 4

[28] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015. 2, 4, 5

[29] F. Shen, C. Shen, W. Liu, and H. Tao Shen. Supervised discrete hashing. In *CVPR*, 2015. 1, 2

[30] C. Strecha, A. Bronstein, M. Bronstein, and P. Fua. Ldahash: Improved matching with smaller descriptors. *PAMI*, 34(1):66–78, 2012. 2

[31] J. Wang, T. Zhang, N. Sebe, H. T. Shen, et al. A survey on learning to hash. *PAMI*, 40(4):769–790, 2018. 2

[32] X. Wang, Y. Shi, and K. M. Kitani. Deep supervised hashing with triplet labels. In *ACCV*, 2016. 1, 4

[33] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, 2009. 1, 4, 5

[34] Y. Wen, K. Zhang, Z. Li, and Y. Qiao. A discriminative feature learning approach for deep face recognition. In *ECCV*, 2016. 2, 4, 8

[35] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan. Supervised hashing for image retrieval via image representation learning. In *AAAI*, 2014. 5

[36] H.-F. Yang, K. Lin, and C.-S. Chen. Supervised learning of semantics-preserving hash via deep convolutional neural networks. *PAMI*, 2017. 1, 2

[37] H.-F. Yang, K. Lin, and C.-S. Chen. Supervised learning of semantics-preserving hash via deep convolutional neural networks. *PAMI*, 40(2):437–451, 2018. 1, 2, 4

[38] T. Yao, F. Long, T. Mei, and Y. Rui. Deep semantic-preserving and ranking-based hashing for image retrieval. In *IJCAI*, 2016. 1, 4

[39] X. Yuan, L. Ren, J. Lu, and J. Zhou. Relaxation-free deep hashing via policy gradient. In *ECCV*, 2018. 1, 5, 6

[40] H. Zhu, M. Long, J. Wang, and Y. Cao. Deep hashing network for efficient similarity retrieval. In *AAAI*, 2016. 5