Image Retrieval with Well-Separated Semantic Hash Centers

Liangdao Wang, Yan Pan*, Hanjiang Lai, Jian Yin

School of Computer Science and Engineering, Sun Yat-sen University
wangld5@mail2.sysu.edu.cn

Abstract. Recently, some point-wise hash learning methods such as CSQ and DPN adapted "hash centers" as the global similarity label for each category and force the hash codes of the images with the same category to get closed to their corresponding hash centers. Although they outperformed other pairwise/triplet hashing methods, they assign hash centers to each class randomly and result in a sub-optimal performance because of ignoring the semantic relationship between categories, which means that they ignore the fact that the Hamming distance between the hash centers corresponding to two semantically similar classes should be smaller than the Hamming distance between the hash centers corresponding to two semantically dissimilar classes. To solve the above problem and generate well-separated and semantic hash centers, in this paper, we propose an optimization approach which aims at generating hash centers not only with semantic category information but also distinguished from each other. Specifically, we adopt the weight of last fully-connected layer in ResNet-50 model as category features to help inject semantic information into the generation of hash centers and try to maximize the expectation of the Hamming distance between each two hash centers. With the hash centers corresponding to each image category, we propose two effective loss functions to learn deep hashing function. Importantly, extensive experiments show that our proposed hash centers and training method outperform the state-of-the-art hash models on three image retrieval datasets.

Keywords: Hash Centers · Semantic Category Information · Image Retrieval.

1 Introduction

Image hashing method is popular in the field of image retrieval for its highly efficient storage ability and retrieval speed with the objective of representing an image using a binary code. Recently, depending on the development of the deep convolution neural network, deep hashing methods have great improvement in terms of learning non-linear hash functions which encode tight hash codes for similar image pairs while encoding hash codes with a large Hamming distance for dissimilar image pairs.

* Corresponding Author

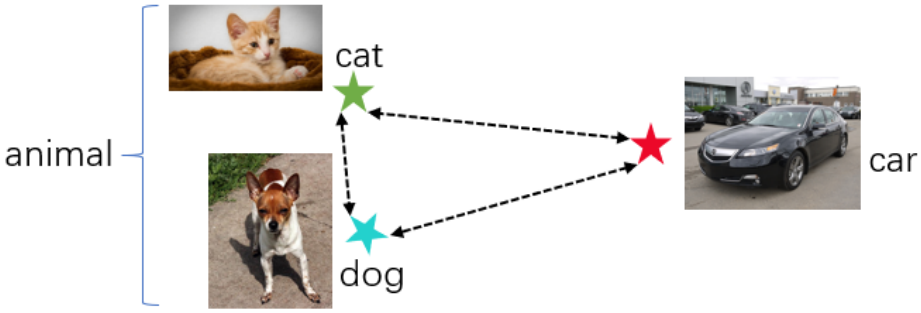


Fig. 1. an example to show that the Hamming distance between cat and dog should be smaller than that between not only dog and car, but also cat and car, because cat and dog are both animal.

Deep hashing methods can be grouped by how the similarity of the learned hashing codes are measured, namely point-wise hashing methods (e.g., [26, 30, 20, 4, 27]), pairwise/triplet hashing methods (e.g., [24, 22, 11, 2]) and listwise hashing methods [28]. Among them, pairwise/triplet hashing methods randomly sample a mini-batch training sample to learn hashing function with pairwise/triplet similarity between these training samples which suffer from three questions: low efficiency in profiling global similarity of the whole dataset, insufficient coverage of data distribution, and low effectiveness on imbalanced data. To solve these issues, some point-wise methods propose to learn hash codes for images with hash centers, by forcing the hash codes of images belonging to the same class to be as similar as the hash centers of their class.

In deep hashing methods with hash centers, we consider that there are two main challenges for learning a good hash function: 1) Firstly, it is important to construct a set of hash centers separated from each other. 2) Secondly, the assignment of hash centers for each class should not be ignored. To solve the above challenges, DPN [4] uses Bernoulli sampling to generate hash centers. CSQ [27] leverages Hadamard matrix and Bernoulli sampling to construct hash centers, considering that Hadamard matrix has a nice property that every two rows are mutually orthogonal. However, both of them choose to assign the generated hash centers randomly to each class, ignoring the fact that the Hamming distance between hash centers corresponding to two categories with similar semantic information should be smaller than that between hash centers corresponding to two categories with irrelevant semantic information. For example, as can be seen in Fig.1, the similarity between dogs and cats should be greater than that between not only dogs and cars, but also cats and cars, because cats and dogs are both animals, but the other two pairs do not belong to the same species. The neglect of semantic relation information between categories makes them get sub-optimal performance.

To address both challenges, we propose a novel deep hashing method that employs an optimization procedure with two optimization targets. The first op-

timization target is to make the expectation of the Hamming distance between each two hash centers as large as possible to address the first challenge. In terms of the second optimization target, we introduce a variable called category feature that contains semantic information of category and makes the inner product between two hash centers should be close to the inner product between their corresponding category features to address the second challenge, which means that the similarity between two hash centers should be closed to that between their corresponding category features.

Our deep hashing method includes a two-stage pipeline. In the first stage, inspired by the existing retrieval/segmentation method([29, 15]), we use the weight of last fully-connected layer in fine-tuned ResNet-50 model as the category features and solve the above optimization problem to obtain the semantic preserved and separated hash centers for each image class. To solve the NP-hard optimization issue, we develop an alternating optimization procedure based on the ℓ_p-box binary optimization scheme. In the second stage, with the constructed hash centers corresponding to each class, we train a deep neural network to learn the hashing function. Specifically, we define loss functions to encourage that (1) the hash code of an input image is nearby the hash center of its class but distanced from other hash centers and (2) quantization errors between continuous code and hash code are minimized during training.

To evaluate our proposed method, we experiment on three image datasets for image retrieval. The results demonstrate that the generated hash centers contain category semantic information of their corresponding class and are separated from each other. The proposed method achieves superior retrieval performance over the state-of-the-art deep hashing methods. The code is relased in https://github.com/Wangld5/SHC-IR

2 Related Work

The traditional hashing methods use hand-craft image features and shallow hash functions to learn hash codes for each input image. For example, SDH [18] presents an SVM-like formulation to minimize the quantization loss and the classification loss to learn hash function. SH [25] is a graph-based hashing method that learns hash function by spectral graph partitioning. Recently, deep hashing methods which learn hash codes for images by DNN have dominated the hashing research due to the superior learning ability of deep neural network and can be divided into point-wise hashing methods [26, 30, 20, 4, 27], pairwise/triplet hashing methods [24, 22, 11, 2, 12, 14, 3], and listwise hashing methods [28].

To keep the similarity structure of data, pairwise/triplet hashing methods learn hash function by forcing the similar pairs of images to have similar hash codes while dissimilar pairs of images to have dissimilar hash codes, or maintaining the consistency between the data triplets from the original and Hamming space. For example, CNNH [24] learn the hash function by forcing the similarity of the image pairs to that of their corresponding approximate binary code pairs. DPSH [12] adopts deep networks to learn hash function with the standard form

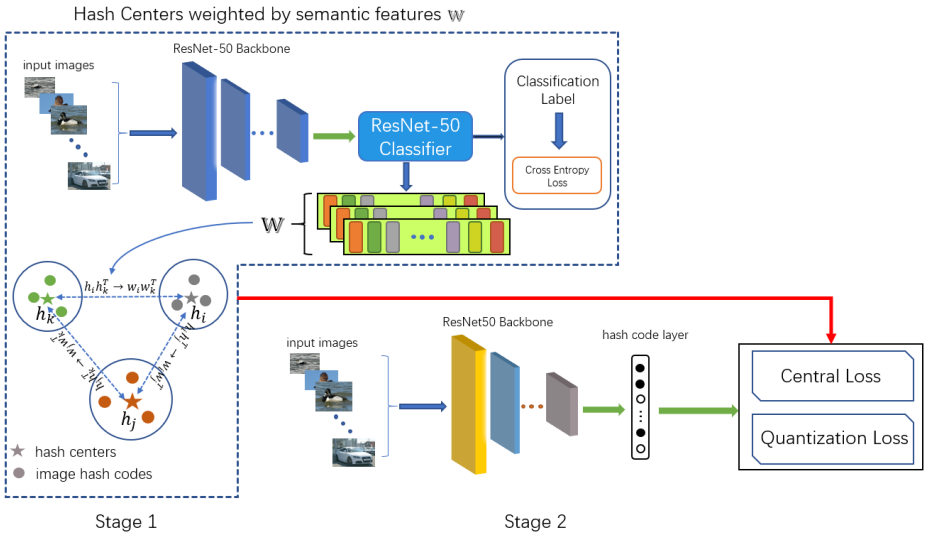


Fig. 2. Overview of the proposed method with a pipeline of two stages. Stage 1 (left) uses a carefully designed optimization procedure to generate hash centers containing semantic information with the help of category features w where each hash center corresponding to an image class and w is the weight of last fully-connected layer obtained from the fine-tuned ResNet-50 classification model. Stage 2 (**right**) uses a deep hashing network trained with two loss functions. The first loss encourages the hash code of an image to become close to its hash center and faraway from others. The second loss reduces quantization errors.

of likelihood loss based on similarity information of the image pairs. HashNet [3] trains the hashing network by adopting maximum likelihood loss with different weights for each image pair.

To solve the three problems of pairwise/triplet hashing methods mentioned in Section 1. Point-wise hashing methods use a target as a guide to learn hash functions. On the one hand, some hashing methods use aware label information for each sample to guide the learning of the hash functions such as DLBHC [13], SSDH [26] and Greedy Hash [20]. On the other hand, some hashing methods generate binary codes called hash centers corresponding to each semantic class in Hamming space which are distinguished from each other and generate hash codes for input images with these separated hash centers as supervised information. For example, DPN [4] proposed to generate hash centers by Bernoulli sampling. CSQ [27] proposed to use Hadamard matrix and Bernoulli sampling to generate a set of hash centers, with the expectation that the average Hamming distance of all pairs of hash centers is half of the code length. However, both CSQ and DPN randomly allocate hash centers, ignoring the semantic relationship between their corresponding categories which may lead to inferior retrieval performance.

Inspired by some unsupervised hashing methods [32, 9] using semantic information, in this work, we propose an optimization approach that generates

hash centers under the guidance of semantic relationships between categories, which make the hash centers not only separated from each other but also contain semantic information.

3 Approach

For the deep hashing problem, given an input image x, hash learning aims to obtain a mapping function $f : x \to b \in \{-1, 1\}^q$ that encodes the image to a binary hash code b with length q for the convenience of image retrieval, so that similar images or images belonging to the same class are mapping to similar hash codes with small Hamming distances. In deep hashing methods, f is a non-linear hashing function implemented by the deep hashing network.

As shown in Fig. 2, to achieve this goal, we propose a deep hashing method of two stages. In Stage 1, based on the initialization of hash centers with Hadamard Matrix, we propose an optimization method to optimize and generate the hash centers with semantic information to ensure that the Hamming distance between two hash centers is associated with the semantic similarity between their corresponding categories. More specifically, we try to decrease the root mean square error between the inner product of two hash centers and that of their corresponding category features in the optimization procedure to increase the semantic information of hash centers, each of which corresponds to one class, respectively. In Stage 2, we train a deep hashing network to learn f with two losses: 1) the central loss to force hash codes of the input images to not only get closed to their corresponding hash centers but also separate from other hash centers with long distance. 2) the quantization loss to decrease errors between continuous codes and binary codes. We present the details of each stage as follows.

3.1 Stage 1: Generate Hash Centers by Optimization

In this stage, we propose an optimization method to find a set of separated hash centers that contain semantic information, which adopts the category feature extracted from the pre-trained models to inject the semantic information into hash centers.

Optimization Target For images in m classes, utilizing the category features $w_1, w_2, ..., w_m$, we try to learn m hash centers $c_1, c_2, ..., c_m$ by maximizing the following optimization target. To measure the similarity between a pair of hash centers and their corresponding category features, we simultaneously enforce the inner product of two hash centers closed to the inner product of their corresponding category features. Specifically, we formulate an optimization objective as:

$$\max_{c_1,...,c_m \in \{-1,1\}^q} \frac{1}{m(m-1)} \sum_i \sum_{j:j \neq i} (||c_i - c_j||_H - ||c_i^T c_j - w_i^T w_j||_2^2) \tag{1}$$

where $||.||_H$ represents the Hamming distance, and $||.||_2$ be the ℓ_2 norm

For two hash centers $c_i, c_j \in \{-1, 1\}^q$, the Hamming distance between them can be transform to the inner product between c_i and c_j with the following equation:

$$||c_i - c_j||_H = \frac{1}{4}||c_i - c_j||_2^2 = \frac{1}{4}(c_i^T c_i + c_j^T c_j - 2c_i^T c_j) = \frac{1}{2}(q - c_i^T c_j) \quad (2)$$

where q is the constant hash code length. Therefore, maximizing $||c_i - c_j||_H$ is equivalent to minimizing $c_i^T c_j$ and the objective in Eq.(1) can be equivalent to:

$$\min_{c_1,...,c_m \in \{-1,1\}^q} \sum_i \sum_{j:j \neq i} (c_i^T c_j + ||c_i^T c_j - w_i^T w_j||_2^2) \quad (3)$$

Obtaining the category feature w For the objective in Eq.(1), we need to obtain the semantic category features w for their corresponding class so that the similarity between any two hash centers c_i, c_j of their corresponding classes get closed to the similarity of their corresponding semantic category feature w_i, w_j. To address this issue, inspired by the previous work [29, 15], we compare three pre-trained models including VGG16 [19], ResNet50 [7], AlexNet [1] (the comparison results can be seen in supplementary material) and choose to fine-tune ResNet-50 pre-trained model on three experiment datasets with cross-entropy loss, and the weight of last fully-connected layer is considered as the category features for each class. However, the range of the inner product of two category features w_i, w_j can not be guaranteed to be the same as the inner product hash centers c_i, c_j which is $[-q, q]$. To solve the problem, we use linear transformation to transform the range of inner products of two category features into $[-q, q]$. Specifically, for two category features w_i, w_j corresponding to hash center c_i, c_j, the transformation formula of the inner product between w_i and w_j can be defined as:

$$w_i^T w_j = -q + 2q \frac{w_i^T w_j - min(w_i^T w_{\sim i})}{max(w_i^T w_{\sim i}) - min(w_i^T w_{\sim i})} \quad (4)$$

where $w_i^T w_j$ represents the inner product between $w_i.w_j$, and $w_{\sim i} = [w_1, ..., w_{i-1}, w_{i+1}, ..., w_m]$ represents the matrix that consists of $w_j (1 \leq j \leq m, j \neq i)$. The derivation process can be seen in the supplementary material.

Alternative Optimization Procedure Since it is NP-hard to optimize the target in Eq. 3 for the binary constraint of hash centers. Alternatively, for each hash center c_i, we adopt an optimization procedure that updates c_i by fixing other centers c_j $(1 \leq j \leq m, j \neq i)$.

Specifically, with all c_j $(j \neq i)$ being fixed, the sub-problem for c_i can be formulated as:

$$\min_{c_i \in \{-1,1\}^q} \sum_{j:j \neq i} (c_i^T c_j + ||h_i^T h_j - w_i^T w_j||_2^2) \quad (5)$$

Inspired by the Proposition 1 in [23], we prove that the binary constraint $z \in \{-1,1\}^q$ is equivalent to $z \in [-1,1]^q \bigcap \{z : ||z||_p^p = q\}$ (The proof can be found in the supplementary material) and adopt the ℓ_p-box algorithm [23] to solve the subproblem in Eq.(5). For ease of optimization, hereafter we set $p = 2$.

With this fact, the binary constraint can be replaced and Eq. 5 can be equivalent to the following form:

$$\begin{aligned} \min_{c_i,z_1,z_2} \quad & \sum_{j:j\neq i} (c_i^T c_j + ||h_i^T h_j - w_i^T w_j||_2^2) \\ s.t. \quad & c_i = z_1,\ c_i = z_2, z_1 \in \mathcal{S}_b,\ z_2 \in \mathcal{S}_p, \end{aligned} \tag{6}$$

where $\mathcal{S}_b = \{z : -1_q < z < 1_q\}$, $\mathcal{S}_p = \{z : ||z||_2^2 = q\}$, 1_q represents a q-dimensional vector with all ones. For ease of solution, the augmented Lagrange function of Eq.(6) is:

$$\begin{aligned} L(c_i, z_1, z_2, y_1, y_2) = & \sum_{j\neq i} (c_i^T c_j + ||c_i^T c_j - w_i^T w_j||_2^2) + y_1^T (c_i - z_1) + y_2^T (c_i - z_2) \\ & + \frac{\mu}{2}||c_i - z_1||_2^2 + \frac{\mu}{2}||c_i - z_2||_2^2 \qquad s.t.\ z_1 \in S_b, z_2 \in S_p, \end{aligned} \tag{7}$$

where y_1, y_2 are Lagrange multipliers.

Then we will perform the following update steps for c_i, z_1, z_2, y_1, y_2 through Alternating Direction Method of Multipliers(ADMM).

Update c_i By fixing other variables except c_i, the sub-problem in Eq. 7 is an unconstrained convex function for c_i. The gradient of Eq. 7 for h_i is

$$\frac{\partial L(c_i)}{\partial c_i} = 2\mu c_i + 2C_{\sim i} C_{\sim i}^T c_i + \sum_{j\neq i} c_j + y_1 + y_2 - \mu(z_1 + z_2) - 2(C_{\sim i} w_{\sim i}^T w_i).$$

Where $C_{\sim i} = [c_1, ..., c_{i-1}, c_{i+1}, ..., c_m]$ represents a matrix that consists of $c_j (1 \leq j \leq m, j \neq i)$. By setting this gradient to zero, we can update c_i by

$$c_i \leftarrow (2\mu I_q + 2C_{\sim i} C_{\sim i}^T)^{-1} (\mu(z_1 + z_2) + 2(C_{\sim i} w_{\sim i}^T w_i) - \sum_{j\neq i} c_j - y_1 - y_2). \tag{8}$$

Update z_1, z_2 The subproblem of L in Eq.(7) for z_1 and z_2 are

$$\begin{cases} L(z_1) = y_1^T (c_i - z_1) + \dfrac{\mu}{2}||c_i - z_1||_2^2 \quad s.t. z_1 \in S_b \\ L(z_2) = y_2^T (c_i - z_2) + \dfrac{\mu}{2}||c_i - z_2||_2^2 \quad s.t. z_2 \in S_p \end{cases} \tag{9}$$

We update z_1 and z_2 by setting the gradient of Eq.9 to be zero:

$$\begin{cases} z_1 \leftarrow P_{S_b}(c_i + \dfrac{1}{\mu} y_1) \\ z_2 \leftarrow P_{S_p}(c_i + \dfrac{1}{\mu} y_2) \end{cases} \tag{10}$$

Algorithm 1 Hash Centers generation process

Input: $C = [c_1, ...c_m]$ initialized by Hadamard Matrix and Bernoulli sampling.
Initialize: learning rate $\rho = 1.03$, iteration number $T = 50$, $\mu_{max} = 10^{10}$, tolerance error $\epsilon = 10^{-6}$, $w_1, ..., w_m$ obtained from ResNet-50 model.

for $t = 1, ..., T$ **do**
  for $i = 1, ..., c$ **do**
    initialize $z_1 = z_2 = c_i$, $y_1 = y_2 = 0_q$, $\mu = 10^6$
    repeat
      update c_i^t by Eq.8
      project and update z_1^t, z_2^t on S_b, S_p with Eq.11
      update y_1^t, y_2^t with gradient ascent by Eq.12
      update $\mu = min(\rho\mu, \mu_{max})$
    until $\max(||c_i^t - z_1^t||_\infty, ||c_i^t - z_2^t||_\infty) \leq \epsilon$
  end for
  t=t+1, calculate L^t by Eq.3
  if the relative change $\frac{L^{t-1} - L^t}{L^{t-1}} \leq \epsilon$, then break
end for

Output: hash centers $C = [c_1, ..., c_m]$

Following [23], we project z_1 and z_2 onto their corresponding space S_b and S_p and use the following closed form solutions to update them:

$$\begin{cases} z_1 \leftarrow \min(1, max(-1, c_i + \dfrac{1}{\mu} y_1)) \\ z_2 \leftarrow \sqrt{\mu q} \dfrac{c_i + y_2}{||c_i + y_2||_2} \end{cases} \tag{11}$$

Update y_1, y_2 We use the conventional gradient ascent to update y_1 and y_2:

$$\begin{cases} y_1 \leftarrow y_1 + \mu(c_i - z_1) \\ y_2 \leftarrow y_2 + \mu(c_i - z_2) \end{cases} \tag{12}$$

The sketch of the proposed optimization procedure is shown in Algorithm 1.

3.2 Stage 2: Train a Deep Hashing Network

As shown in the right part of Fig. 2, we build a deep hashing network to learn hash functions with the generated semantic hash centers as label information. The hashing network can be divided into two blocks.

The first block is a backbone consisting of multiple convolution layers based on ResNet-50 [7] following a hashing code layer consisting of a fully connected layer and *Tanh* activation function. The goal of the first block is to generate continuous codes $v \in [-1, 1]^q$ for each input image during the training process. In the testing process, each continuous code v will be converted to binary hash code $b \in \{-1, +1\}^q$ by quantization. Specifically, we define that the training set has m hash centers $c_1, c_2, ..., c_m$, N image $I_1, I_2, ..., I_N$ whose output continuous codes are $v_1, v_2, ..., v_N$, respectively,

The second block is a two-loss function served for the training of hashing network. One is to encourage the continuous codes of images belonging to the same class nearby their corresponding hash centers while separated from other hash centers with a high distance. While another is to reduce the quantization loss between continuous codes and binary hash codes. We will introduce the details of these losses, respectively.

Central Loss With the obtained m hash centers from Stage 1, we assign the hash center to each image class according to the category weights w. We develop a loss function that forces the continuous code of an image to approach the hash center assigned to this image's class while being faraway from the hash centers of other categories. Specifically, for the image I_j, the loss function that measures the error between the continuous code v_j of I_j and its corresponding hash center c_i can be defined as:

$$L_{C,j,i} = -(\log P_{j,i} + \frac{1}{m-1}\sum_{k\neq i}^{m}\log(1-N_{j,k})) \tag{13}$$

with

$$P_{j,i} = \frac{e^{-S(v_j,c_i)}}{\sum_{p=1}^{m} e^{-S(v_j,c_p)}}, N_{j,k} = \frac{e^{-S(v_j,c_k)}}{\sum_{p=1}^{m} e^{-S(v_j,c_p)}} \tag{14}$$

where $S(x,y)$ represents the similarity metric between x and y, $P_{j,i}$ represents the probability that the continuous code v_j of image I_j belongs to its corresponding hash center c_i while $N_{j,k}$ represents the probability that the continuous code v_j of image I_j belongs to other hash centers of different categories which should be reduced. Following the existing hashing methods [5, 8], we use the scaled cosine similarity as the similarity metric, so $P_{j,i}$ in Eq.(14) can be reformulated as:

$$P_{j,i} = \frac{e^{\sqrt{q}\cos(v_j,c_i)}}{\sum_{p=1}^{m} e^{\sqrt{q}\cos(v_j,c_p)}}, N_{j,k} = \frac{e^{\sqrt{q}\cos(v_j,c_k)}}{\sum_{p=1}^{m} e^{\sqrt{q}\cos(v_j,c_p)}}, \tag{15}$$

where $\cos(x,y) = \frac{x^T y}{||x||_2||y||_2}$ represents the cosine similarity between the vectors x and y, $\sqrt{q}$ is the scale factor with q being the hash code length.

Quantization loss To make it easy to back propagate the gradient of the loss function, the continuous outputs of the deep hashing network are used as a relaxation of the binary hash codes during training. To reduce quantization error between binary hash codes and continuous outputs, similar to existing methods (e.g. [31]), the quantization loss of continuous code v_j is defined as:

$$L_{Q,j} = |||v_j| - 1_q||_1, \tag{16}$$

where $||.||_1$ is the ℓ_1 norm, 1_q represents a q-dimensional vector with all ones.

Table 1. Comparison in mAP of Hamming Ranking for different bits on image retrieval.

Method	ImageNet (mAP@1000)			NABirds (mAP@All)			Stanford Cars (mAP@All)		
	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
SH [25]	0.1705	0.2997	0.4561	0.0107	0.0200	0.0312	0.0130	0.0160	0.0210
ITQ-CCA [6]	0.1907	0.3850	0.5325	0.0135	0.0270	0.0452	0.0163	0.0235	0.0323
SDH [18]	0.3416	0.4956	0.5990	0.0102	0.0225	0.0459	0.0161	0.0231	0.0298
DSH [14]	0.7179	0.7448	0.7585	0.0820	0.1011	0.2030	0.2153	0.3124	0.4309
DPSH [12]	0.6241	0.7626	0.7992	0.1171	0.1855	0.2811	0.1764	0.2949	0.4132
HashNet [3]	0.6024	0.7158	0.8071	0.0825	0.1439	0.2359	0.2637	0.3611	0.4845
GreedyHash [20]	0.7394	0.7977	0.8243	0.3519	0.5350	0.6117	0.7312	0.8271	0.8432
DPN [4]	0.7987	0.8298	0.8394	0.6151	0.6928	0.7244	0.7287	0.8214	0.8488
CSQ [27]	0.8377	0.8750	0.8836	0.6183	0.7210	0.7491	0.7435	0.8392	0.8634
OrthoHash [8]	0.8540	0.8792	0.8936	0.6366	0.7243	0.7544	0.8012	0.8490	0.8676
Ours	**0.8616**	**0.8851**	**0.8982**	**0.6693**	**0.7381**	**0.7599**	**0.8218**	**0.8569**	**0.8771**

Combination of Loss Functions We combine the three loss functions to form the optimization objective used in the proposed deep hashing network.

$$L = \frac{1}{N}\sum_{j=1}^{N} L_{C,j,i} + \alpha \frac{1}{N}\sum_{j=1}^{N} L_{Q,j} \tag{17}$$

where α are trade-off hyper-parameters.

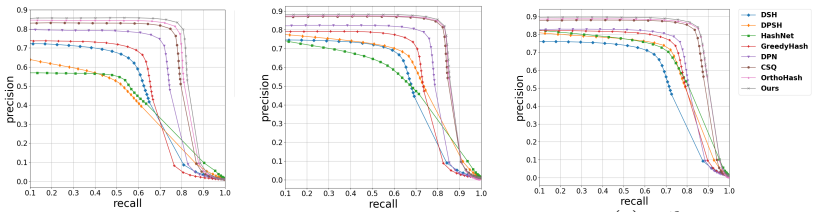
4 Experiments

4.1 Experiment Settings

Three benchmark image dataset for image retrieval are used to conducted our experiments, including, ImageNet [17], NABirds [21] and Stanford Cars [10]. Following the same settings as in [3, 27], the ImageNet dataset we used consists of 143,495 images for 100 classes, including 100 images for each class as training set, 50 images for each class as test queries, and the rest images as the retrieval database. For Stanford Cars containing 16,185 images in 196 classes and NABirds containing 48,562 images in 555 classes, the training set in the official train/test split is used as the training images and the retrieval database, the test set in the official train/test split as the test queries for retrieval. The identical training set, test set and retrieval database are applied to our method and all other comparison methods for a fair comparison.

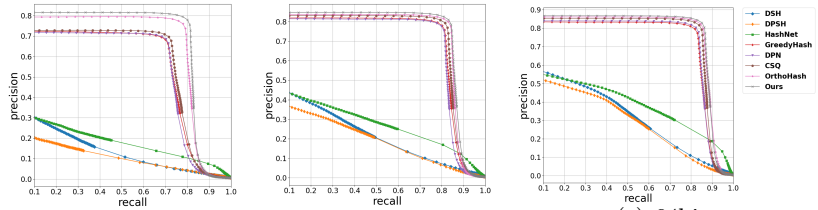
We compare the performance of the proposed method with nine baselines, including six popular deep hashing methods OrthoHash [8], CSQ [27], DPN [4], Greedy Hash [20], HashNet [3], DPSH [12] and DSH [14], three conventional hashing methods SH [25], ITQ-CCA [6] and SDH [18]. For the three conventional hashing methods, we adopt the output of the last fully-connected layer in the pre-trained ResNet-50 model as the input features. For a fair comparison, the pre-trained ResNet-50 model is used as the backbone for all deep hashing methods.

Two widely-used evaluation metrics are used to evaluate the effectiveness of our method, including Mean Average Precision (MAP) and Precision-Recall curves. Following [3, 27], we use MAP@1000 as the MAP metric on ImageNet because it contains a large-scale retrieval database. On NABirds and Stanford Cars, we use MAP@ALL as the MAP metric.



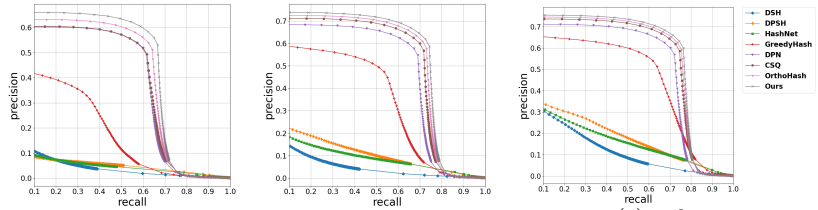
(a) 16bits (b) 32bits (c) 64bits

Fig. 3. Comparison results w.r.t. Precision-Recall curves on ImageNet



(a) 16bits (b) 32bits (c) 64bits

Fig. 4. Comparison results w.r.t. Precision-Recall curves on Stanford Cars



(a) 16bits (b) 32bits (c) 64bits

Fig. 5. Comparison results w.r.t. Precision-Recall curves on NABirds

4.2 Implementation Details

We implement our methods with Pytorch [16]. For all experiments, the optimizer of the deep hashing network in Stage 2 is RMSprop(root mean square prop) with the initial learning rate $lr = 10^{-5}$ and the batch size is set as 64. To determine the proper hyper-parameter α in the loss function, we randomly split the official training set into a validation set with 20% images and

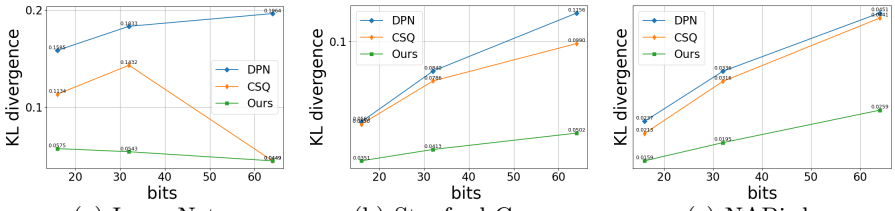
a sub-training set with the rest 80% images for each dataset. Then we determine α through the validation set after the model is trained on the sub-training set. Finally, we use the obtained α to train the model on the official training set. In the testing process, each image will be encoded into a hash code with $b = sign(v)$, where v is the output continuous code of our deep hashing network.

Table 2. Comparison results of the mean Hamming distance and minimum Hamming distance over all pairs of hash centers, for different ways to generate hash centers

Datasets	Methods	16bits		32bits		64bits	
		min	mean	min	mean	min	mean
ImageNet	DPN Centers	2	7.96	6	15.98	18	32.02
	CSQ Centers	2	8.04	6	16.11	32	32.23
	Ours Centers	3	8.08	10	16.15	32	32.23
Stanford Cars	DPN Centers	0	7.97	6	15.99	16	32.02
	CSQ Centers	0	8.01	6	16.03	16	32.10
	Ours Centers	3	8.05	9	16.08	22	32.15
NABirds	DPN Centers	0	7.96	4	15.95	12	31.98
	CSQ Centers	0	7.98	4	15.98	14	32.01
	Ours Centers	2	8.01	8	16.03	20	32.05

Table 3. Comparison results of the Mean Average Precision (MAP) for different ways to generate hash centers on three image datasets with the same settings of Stage 2

Datasets	Methods	16bits	32bit	64bits
ImageNet	DPN Center	0.8218	0.8501	0.8876
	CSQ Center	0.8498	0.8787	0.8982
	Ours Center	0.8616	0.8851	0.8982
Stanford Cars	DPN Center	0.7611	0.8378	0.8602
	CSQ Center	0.7790	0.8475	0.8681
	Ours Center	0.8218	0.8569	0.8771
NABirds	DPN Center	0.6309	0.7090	0.7448
	CSQ Center	0.6335	0.7136	0.7527
	Ours Center	0.6693	0.7381	0.7599



(a) ImageNet (b) Stanford Cars (c) NABirds

Fig. 6. The KL divergence of the distribution of similarity between hash centers relative to the distribution of similarity between category features on three experiment datasets with different length of hash centers

4.3 Results of Retrieval Accuracies

Table 1 shows the Mean Average Precision (MAP) results of retrieval performance on three datasets. On all of these datasets, the proposed method achieves superior retrieval accuracies against the baseline methods. For example, compared to the best baseline, the MAP results of the proposed method indicate a relative increase of **0.51**% ∼**0.89**% / **1.1**% ∼ **2.6**% / **0.73**% ∼ **5.1**% on ImageNet / Stanford Cars / NABirds, respectively. Fig.3, Fig.4, and Fig.5 shows the retrieval performance in Precision-Recall curves (P-R curve) on three datasets. The comparison above shows that the well-separated hash centers with semantic information we generated can effectively improve the retrieval effect. Note

that the proposed method makes large improvements when the number of image classes is large and the length of hash code is short. For example, when the hash code is 16-bits, on Stanford Cars with 196 classes or NABirds with 555 classes, the MAP results of the proposed method show a relative improvement of **2.6**% or **5.1**% compared to the best baseline OrthoHash, respectively. The reason may be that when the code length is short, previous methods will generate two hash centers with Hamming distance of 0 and contain no semantic information which make the performance to be sub-optimal. But, with the injection of semantic relevance, the proposed method not only generates well-separated hash centers but also contains semantic information.

4.4 Ablation Studies

Effectiveness of Hash Centers The main contribution of this paper is the proposed optimization procedure to generate mutually separated hash centers with semantic information. To verify the effectiveness of the generated hash centers by the proposed method, we adopt two existing methods to generate hash centers as baselines. The two baselines to be compared are 1) center learning in CSQ, and 2) center learning in DPN. We train a deep hashing network with these hash centers by using the same settings as the proposed method in Stage 2. The only difference between these baselines and the proposed method lies in different ways to generate hash centers.

To explore how much category semantic relation information is contained in the hash centers generated by not only the proposed method but also baselines, we calculate the KL divergence of the distribution of similarity between hash centers relative to the distribution of similarity between category features, where the similarity between hash centers is expressed by the inner product of them, and so is the similarity between category features. The results in Fig 6 shows that in most case, the KL divergence of the similarity distribution between hash centers generated by our method relative to the similarity distribution between category features is the smallest which means the hash centers generated by our method contains more category semantic relation information.

To explore how well the hash centers generated by different methods are separated from each other, we compare the average of the Hamming distance over all pairs of hash centers and the minimum Hamming distance between any two hash centers for both our method and two baselines. The results shown on Tab. 2 can be observed that (1) In most cases, the minimum Hamming distance of our method is larger than that of the two baselines, which indicates that the baselines can not generate well-separated hash centers in some cases. For example, the minimum Hamming distance between any two hash centers is 0 with 16 bits on both Stanford Cars and NABirds. (2)Compared to both the baselines, the hash centers generated by our methods have largest average Hamming distance. The comparison results show that our method can generate well-separated hash centers with the help of category features. As shown in Tab 3, on all of the datasets, the proposed method outperforms the baseline with a clear gap, e.g., for 16-bit codes on NABirds and Stanford Cars. In summary, the

Table 4. Comparison results w.r.t. MAP for different combinations of loss functions

		ImageNet			**Stanford CAR**			**NABirds**		
L_C	L_Q	16bits	32bits	64bits	16bits	32bits	64bits	16bits	32bits	64bits
✓	✓	0.8616	0.8851	0.8982	0.8218	0.8569	0.8771	0.6693	0.7381	0.7599
✓		0.8572	0.8828	0.8958	0.8048	0.8531	0.8671	0.6526	0.7236	0.7539
CSQ-L_c		0.8543	0.8767	0.8836	0.7761	0.8481	0.8739	0.6429	0.7202	0.7464
DPN-L		0.8323	0.8573	0.8677	0.7643	0.8337	0.8503	0.6604	0.7093	0.7383

ablation results in Fig.6, Tab 2 and 3 verify the hash centers obtained by our optimization approach, which not only contain category semantic information but also are well separated, can help to improve the retrieval performance.

Effectiveness of Loss Functions To explore the individual effect of each part of the loss function, we evaluate different combinations of the central loss L_C and the quantization loss L_Q used in the proposed deep hashing network.

In Table 4, with identical hash centers and identical network architecture, we compare the MAP results of three methods. CSQ-L_C is a baseline that uses the loss L_C for hash centers proposed in CSQ [27]. DPN-L is a baseline that uses the central loss for hash centers proposed in DPN [4]. L_C represents only the central loss L_C used for experiments. Two observations can be seen from Table 4: (1) The method with the proposed central loss L_C outperforms both CSQ-L_C and DPN-L. Because we not only encourage hash codes to be close to their corresponding hash centers, but also to be far away from other hash centers. (2) The loss L_Q can also improve the retrieval performance, because we make the continuous codes get closed to the binary hash codes during training.

5 Conclusion

In this paper, we developed an optimization approach to generate well-separated hash centers with semantic information, where we adopt the weight of last fully-connected layer in fine-tuned ResNet-50 model as the category features and force the inner product between a pair of hash centers closed to the inner product between their corresponding category features. With these hash centers, each corresponding to one image class, we propose several effective loss functions to train deep hashing networks. Empirical evaluations in image retrieval show that the proposed method has superior performance gain over state-of-the-arts. In the future, we will continue to explore how to design an effective loss function that better matches the hash center to improve the effect of image retrieval.

Acknowledgements This work was supported in part by the National Science Foundation of China under Grant U1811262, Grant 61772567.

References

1. Alex, K., Ilya, S., Hinton, G.E.: Imagenet Classification with Deep Convolutional Neural Networks. Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS-12) p. 1097–1105 (2012)
2. Cao, Y., Long, M., Liu, B., Wang, J.: Deep Cauchy Hashing for Hamming Space Retrieval. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-18) pp. 1229–1237 (2018)
3. Cao, Z., Long, M., Wang, J., Yu, P.S.: HashNet: Deep Learning to Hash by Continuation. Proceedings of the IEEE International Conference on Computer Vision (ICCV-17) pp. 5608–5617 (2017)
4. Fan, L., Ng, K., Ju, C., Zhang, T., Chan, C.S.: Deep Polarized Network for Supervised Learning of Accurate Binary Hashing Codes. Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence (IJCAI-20) pp. 825–831 (2020)
5. Gong, Y., Kumar, S., Verma, V., Lazebnik, S.: Angular Quantization-based Binary Codes for Fast Similarity Search. Advances in Neural Information Processing Systems (NIPS-12) pp. 1196–1204 (2012)
6. Gong, Y., Lazebnik, S., Gordo, A., Perronnin, F.: Iterative Quantization: A Procrustean Approach to Learning Binary Codes for Large-Scale Image Retrieval. IEEE Transactions on Pattern Analysis and Machine Intelligence **35**(12), 2916–2929 (2013)
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-16) pp. 770–778 (2016)
8. Hoe, J.T., Ng, K.W., Zhang, T., Chan, C.S., Song, Y.Z., Xiang, T.: One Loss for All: Deep Hashing with a Single Cosine Similarity based Learning Objective. Advances in Neural Information Processing Systems (NIPS-21) (2021)
9. Karaman, S., Lin, X., Hu, X., Chang, S.F.: Unsupervised Rank-Preserving Hashing for Large-Scale Image Retrieval. Proceedings of the 2019 on International Conference on Multimedia Retrieval pp. 192–196 (2019)
10. Krause, J., Stark, M., Deng, J., Fei-Fei, L.: 3D Object Representations for Fine-Grained Categorization. Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCV-13) pp. 554–561 (2013)
11. Li, Q., Sun, Z., He, R., Tan, T.: Deep Supervised Discrete Hashing. Advances in Neural Information Processing Systems (NIPS-17) pp. 2479–2488 (2017)
12. Li, W., Wang, S., Kang, W.: Feature Learning Based Deep Supervised Hashing with Pairwise Labels. Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16) pp. 1711–1717 (2016)
13. Lin, K., Yang, H.F., Hsiao, J.H., Chen, C.S.: Deep Learning of Binary Hash Codes for Fast Image Retrieval. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops pp. 27–35 (2015)
14. Liu, H., Wang, R., Shan, S., Chen, X.: Deep Supervised Hashing for Fast Image Retrieval. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-16) pp. 2064–2072 (2016)
15. Long, J., Shelhamer, E., Darrell, T.: Fully Convolutional Networks for Semantic Segmentation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition pp. 3431–3440 (2015)
16. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito,

Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: Advances in Neural Information Processing Systems (NIPS-19). p. 8024–8035. Curran Associates, Inc. (2019)
17. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision **115**(3), 211–252 (2015)
18. Shen, F., Shen, C., Liu, W., Tao Shen, H.: Supervised Discrete Hashing. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-15) pp. 37–45 (2015)
19. Simonyan, K., Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition. ICLR (2015)
20. Su, S., Zhang, C., Han, K., Tian, Y.: Greedy Hash: Towards Fast Optimization for Accurate Hash Coding in CNN. Advances in Neural Information Processing Systems (NIPS-18) pp. 806–815 (2018)
21. Van Horn, G., Branson, S., Farrell, R., Haber, S., Barry, J., Ipeirotis, P., Perona, P., Belongie, S.: Building a Bird Recognition App and Large Scale Dataset With Citizen Scientists: The Fine Print in Fine-Grained Dataset Collection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-15) pp. 595–604 (2015)
22. Wang, X., Shi, Y., Kitani, K.M.: Deep Supervised Hashing with Triplet Labels. Asian Conference on Computer Vision (ACCV-16) pp. 70–84 (2016)
23. Wu, B., Ghanem, B.: ℓ_p-Box ADMM: A Versatile Framework for Integer Programming. IEEE Transactions on Pattern Analysis and Machine Intelligence **41**(7), 1695–1708 (2019)
24. Xia, R., Pan, Y., Lai, H., Liu, C., Yan, S.: Supervised Hashing for Image Retrieval via Image Representation Learning. Proceedings of the AAAI conference on Artificial Intelligence (AAAI-14) pp. 2156–2162 (2014)
25. Yair, W., Antonio, T., Robert, F.: Spectral Hashing. Advances in Neural Information Processing Systems (NIPS-08) pp. 1753–1760 (2008)
26. Yang, H.F., Lin, K., Chen, C.S.: Supervised Learning of Semantics-Preserving Hash via Deep Convolutional Neural Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence **40**(2), 437–451 (2017)
27. Yuan, L., Wang, T., Zhang, X., Tay, F.E., Jie, Z., Liu, W., Feng, J.: Central Similarity Quantization for Efficient Image and Video Retrieval. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-20) pp. 3083–3092 (2020)
28. Zhao, F., Huang, Y., Wang, L., Tan, T.: Deep Semantic Ranking Based Hashing for Multi-Label Image Retrieval. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition pp. 1556–1564 (2015)
29. Zheng, X., Ji, R., Sun, X., Zhang, B., Wu, Y., Huang, F.: Towards Optimal Fine Grained Retrieval via Decorrelated Centralized Loss with Normalize-scale Layer. Proceedings of the AAAI Conference on Artificial Intelligence pp. 9291–9298 (2019)
30. Zhou, C., Po, L.M., Yuen, W.Y., Cheung, K.W., Xu, X., Lau, K.W., Zhao, Y., Liu, M., Wong, P.H.: Angular Deep Supervised Hashing for Image Retrieval. IEEE Access **7**, 127521–127532 (2019)
31. Zhu, H., Long, M., Wang, J., Cao, Y.: Deep Hashing Network for Efficient Similarity Retrieval. Proceedings of the AAAI conference on Artificial Intelligence (AAAI-16) pp. 2415–2421 (2016)

32. Zieba, M., Semberecki, P., El-Gaaly, T., Trzcinski, T.: BinGAN: Learning Compact Binary Descriptors With a Regularized GAN. Advances in Neural Information Processing Systems **31** (2018)