Deep Hashing with Minimal-Distance-Separated Hash Centers

Liangdao Wang[1], Yan Pan[1]*, Cong Liu[1], Hanjiang Lai[1], Jian Yin[1], Ye Liu[1,2]
[1]School of Computer Science and Engineering, Sun Yat-Sen University. [2]Big Data Department, Lizhi Inc.
wangld5@mail2.sysu.edu.cn,panyan5@mail.sysu.edu.cn

Abstract

Deep hashing is an appealing approach for large-scale image retrieval. Most existing supervised deep hashing methods learn hash functions using pairwise or triple image similarities in randomly sampled mini-batches. They suffer from low training efficiency, insufficient coverage of data distribution, and pair imbalance problems. Recently, central similarity quantization (CSQ) attacks the above problems by using "hash centers" as a global similarity metric, which encourages the hash codes of similar images to approach their common hash center and distance themselves from other hash centers. Although achieving SOTA retrieval performance, CSQ falls short of a worst-case guarantee on the minimal distance between its constructed hash centers, i.e. the hash centers can be arbitrarily close. This paper presents an optimization method that finds hash centers with a constraint on the minimal distance between any pair of hash centers, which is non-trivial due to the non-convex nature of the problem. More importantly, we adopt the Gilbert-Varshamov bound from coding theory, which helps us to obtain a large minimal distance while ensuring the empirical feasibility of our optimization approach. With these clearly-separated hash centers, each is assigned to one image class, we propose several effective loss functions to train deep hashing networks. Extensive experiments on three datasets for image retrieval demonstrate that the proposed method achieves superior retrieval performance over the state-of-the-art deep hashing methods.

1. Introduction

Hashing methods are widely-used in large-scale image retrieval due to their excellent efficiency in both storage and retrieval. Recently, much effort has been devoted to deep-learning-based hashing (**deep hashing**) methods for image retrieval. They use deep neural networks to learn hash functions that encode similar/dissimilar images to nearby/faraway binary codes, respectively. Most of the existing deep hashing methods train models on pairwise/triple similarities among training samples in randomly sampled mini-batches (e.g., [1, 10, 18, 22, 24]). Very recently, Yuan *et al.* [26] pointed out that these methods lead to restricted performance due to three problems: low-efficiency to obtain global similarity of the dataset, incomplete coverage of data distribution that harms the discriminability of the generated hash codes, and ineffectiveness on imbalanced amount of similar/dissimilar data pairs. They then proposed central similarities that finds mutually separated hash centers for each class of similar images, and uses these centers to ensure small distances between the hash codes of similar images and large distances between those of dissimilar ones.

For deep hashing methods that use hash centers, it is crucial to construct well-separated hash centers, i.e. the Hamming distance between two hash centers should be significantly larger than the Hamming distance between the hash codes of two similar images, which makes it challenging to generalize to various length of hash code and different number of image classes. For instance, CSQ [26] adopts Hadamard matrix and Bernoulli sampling to produce hash centers with nice properties that any two centers' Hamming distance is on average half of the hash code length. However, pairs of hash centers constructed in this way can be arbitrarily small in the worst case, i.e. zero in Hamming distance (see Table 4). These degenerated hash centers are expected to harm the retrieval performance.

To address this issue, we propose a novel deep hashing method that uses an optimization procedure to produce hash centers, with an additional constraint on a given minimal distance d between any pair of hash centers. The value of d is derived using the Gilbert-Varshamov bound [20] adopted from coding theory, which help us to find a large d while ensuring the feasibility of our optimization procedure.

As shown in Fig.1, the proposed method employs a two-stage pipeline. In Stage 1, we tackle the optimization problem stated above to produce clearly-separated hash centers. To solve this optimization problem, we propose an alternating optimization procedure that relies on the ℓ_p-box binary optimization technique [23]. In Stage 2, we train a deep hashing network by using the constructed hash centers as a

*Corresponding Author

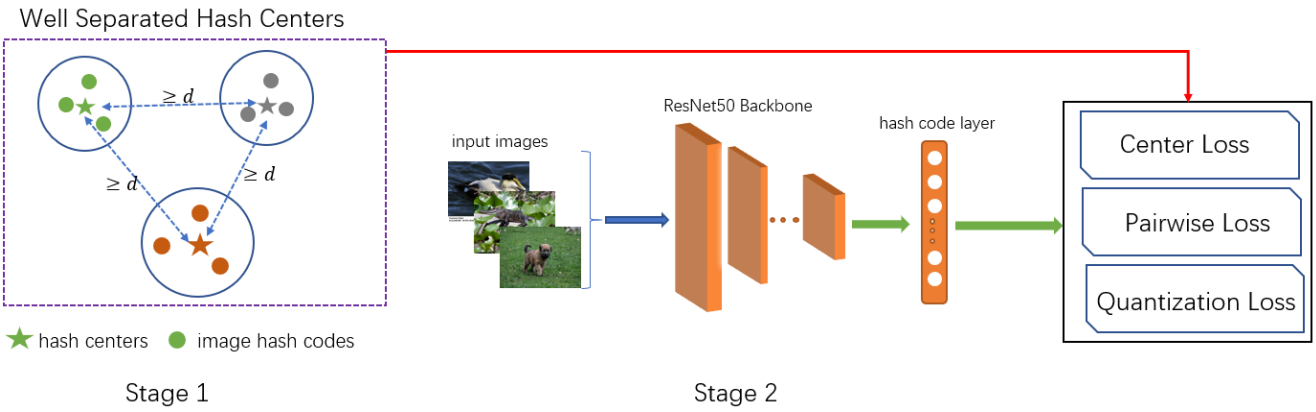


Figure 1. The proposed method comprises of a two-stage pipeline. Stage 1 (**left**) employs a optimization procedure to produce hash centers constrained by a minimal Hamming distance d between any pair of hash centers, each is assigned to one image class. d is given by the Gilbert-Varshamov bound that guarantees the optimization's feasibility. Stage 2 (**right**) employs a deep hashing network with three loss functions. The first one brings the hash code of an image close to its corresponding hash center while keeping it distant from the other centers. The second one draws similar data points within the same class even closer. The last one is to minimize quantization errors.

global similarity metric. Specifically, loss functions are defined to make that (1) an input image's hash code is close to its class's hash center but is distanced from other centers, (2) the hash codes of images in the same class should be close to each other, and (3) quantization errors are minimized.

The proposed method is assessed on three datasets for image retrieval. The results indicate that the obtained hash centers are always separated by the minimal distance we derived, and the proposed method outperforms the state-of-the-art deep hashing methods.

2. Related Work

This paper is built on the data efficient pointwise deep hashing, which we improve by guaranteing a theoretical minimal distance between hash centers and by showing empirical performance gains.

Prior hashing methods for image retrieval [5, 14, 15] usually use hand-crafted visual features as image representations, which are followed by projection and/or quantization to generate hash codes. In recent years, deep-learning-based hashing (**deep hashing**) methods [1, 10, 18, 21, 22, 24] gain popularity in generating more effective and compact binary representations. Deep hashing methods can be roughly divided into pairwise methods, triplet-based methods and pointwise methods.

Pairwise methods [2, 11, 13] learn hash functions by using pairwise similarities between data pairs while triplet based methods [9, 22] leverage relative similarities amongst data triplets. However, both pairwise and triplet-based methods can only capture partial similarity relationships of data, which may suffer from the low training efficiency, insufficient coverage of data distribution, and positive/negative pair imbalance problems [26].

Pointwise methods learn hash functions using the labels of the data points as supervision. Pointwise methods can be further divided into those assuming the availability of class labels [12, 18, 25], and those deriving hash centers [3, 7, 26], where data points in the same semantic class are assigned the same hash center. A key property of the latter is that the distances between any two hash centers should be sufficiently large such that data points of distant semantic class are clearly separable. DPN [3] proposes an optimization algorithm to obtain hash centers. CSQ [26] produces hash centers distanced on average by half of the hash-code length, by using Hadamand matrix and Bernoulli sampling. OrthoHash [7] employs a similar way to CSQ to get hash centers and use them in a single objective. However, hash centers derived in these methods can be arbitrarily close in the worst-case.

Since the generated hash centers may degrade retrieval performance, we propose an optimization approach that guarantees a minimal distance between any pair of hash centers. Specifically, we adopt the Gilbert-Varshamov bound from coding theory, which helps us to obtain a large minimal distance while ensuring the empirical feasibility of our optimization approach.

3. The Proposed Approach

The goal of hash learning for images is to find a function $M : \mathcal{X} \to \{-1, 1\}^q$ that converts an image $x \in \mathcal{X}$ to a hash code in $\{-1, 1\}^q$ with length q, so that similar images have nearby hash codes but dissimilar images have faraway codes. For deep hashing methods, M is usually implemented by a deep network with a layer that outputs hash codes.

The proposed deep hashing method employs a two-stage

pipeline. In Stage 1, we develop an optimization procedure to generate a set of **hash centers** with a constraint that, in the worst case, the Hamming distance between any two centers is not less than a minimal distance d. More importantly, we use the Gilbert-Varshamov bound [20] to determine a large minimal distance d while ensuring the feasibility of our optimization procedure. Each hash center is assigned to one image class.

In Stage 2, we train a deep hashing network with three loss functions: a loss towards hash centers to encourage the hash code of an input image to be nearby the hash center of its class but be far away from other centers, a loss for similar images to encourage the hash codes of the images in the same class to be mutually close, and a loss to reduce quantization errors. In the following, we will illustrate both stages, respectively.

3.1. Stage 1: Generate Hash Centers by Optimization

In Stage 1, we develop an optimization procedure to find a set of hash centers that are mutually separated at least by a minimal distance d, where we adopt Gilbert-Varshamov bound [20] to obtain a large d while ensuring the feasibility of optimization. Each of these hash centers will be assigned to an image class.

3.1.1 Optimization Objective

For images in c classes, we try to learn c hash centers $h_1, h_2, ..., h_c$, which aims to maximize the averaged distance between every two hash centers, and to enforce the Hamming distance between any two centers not less than a minimal distance d (see Section 3.1.3 for details of this minimal distance). The optimization objective is formulated as:

$$\begin{aligned} &\max_{h_1,...,h_c \in \{-1,1\}^q} \frac{1}{c(c-1)} \sum_i \sum_{j:j\neq i} ||h_i - h_j||_H \\ &s.t.\ ||h_i - h_j||_H \geq d\ (1 \leq i, j \leq c, i \neq j), \end{aligned} \tag{1}$$

where $||.||_H$ is the Hamming distance, and d is the minimal distance parameter.

Let $||.||_2$ be the ℓ_2 norm. Since $h_i, h_j \in \{-1, 1\}^q$, we can verify that $4||h_i - h_j||_H = ||h_i - h_j||_2^2 = h_i^T h_i + h_j^T h_j - 2h_i^T h_j = 2q - 2h_i^T h_j$, where q is the hash code length. Hence, maximizing $||h_i - h_j||_H$ is equivalent to minimizing $h_i^T h_j$, $||h_i - h_j||_H \geq d$ is equivalent to $h_i^T h_j \leq q - 2d$. With these facts, the objective in Eq.(1) can be simplified to its equivalent form:

$$\begin{aligned} &\min_{h_1,...,h_c \in \{-1,1\}^q} \sum_i \sum_{j:j\neq i} h_i^T h_j \\ &s.t.\ h_i^T h_j \leq q - 2d\ (1 \leq i, j \leq c, i \neq j). \end{aligned} \tag{2}$$

3.1.2 Alternative Optimization Procedure

The optimization objective in Eq.(2) is a NP-hard problem in general, due to the binary constraints $h_1, h_2, ..., h_c \in \{-1, 1\}^q$. We adopt a procedure that alternatively updates one of the hash centers h_i with other centers h_j $(1 \leq j \leq c, j \neq i)$ being fixed.

By fixing all h_j $(j \neq i)$, we formulate the subproblem w.r.t. h_i as:

$$\begin{aligned} &\min_{h_i \in \{-1,1\}^q} \sum_{j:j\neq i} h_i^T h_j \\ &s.t.\ h_i^T h_j \leq q - 2d\ (1 \leq j \leq c, j \neq i). \end{aligned} \tag{3}$$

We adopt the ℓ_p-box algorithm for binary optimization [23] to solve the subproblem in Eq.(3).

Similarly to Proposition 1 in [23], the binary constraint $v \in \{-1, 1\}^q$ is equivalent to $v \in [-1, 1]^q \bigcap \{v : ||v||_p^p = q\}$. For simplicity, we set $p = 2$.

With this fact, we can drop the binary constraint and reformulate Eq.(3) to the following equivalent form:

$$\begin{aligned} &\min_{h_i, v_1, v_2} \sum_{j:j\neq i} h_i^T h_j \\ &s.t.\ h_i^T h_j \leq q - 2d\ (1 \leq j \leq c, j \neq i) \\ &\quad h_i = v_1,\ h_i = v_2, v_1 \in \mathcal{V}_{box},\ v_2 \in \mathcal{V}_{sph}, \end{aligned} \tag{4}$$

where $\mathcal{V}_{box} = \{v : -1_q < v < 1_q\}$, $\mathcal{V}_{sph} = \{v : ||v||_2^2 = q\}$, 1_q represents a q-dimensional vector whose elements are all ones.

By introducing an auxiliary variable v_3, the inequality constraints $h_i^T h_j \leq q - 2d$ can be replaced by an equality constraint $h_i^T H_{\sim i} + v_3 = (q - 2d)1_{c-1}$ with a simple range constraint $v_3 \in R_+^{c-1}$, where $H_{\sim i} = [h_1, ..., h_{i-1}, h_{i+1}, ..., h_c]$ represents a matrix that consists of $h_j (1 \leq j \leq c, j \neq i)$, $R_+^{c-1} = \{v : v \in [0, +\infty)^{c-1}\}$. The problem in Eq.(4) can be further reformulated as:

$$\begin{aligned} &\min_{h_i, v_1, v_2, v_3} \sum_{j:j\neq i} h_i^T h_j \\ &s.t.\ h_i^T H_{\sim i} + v_3 = (q - 2d)1_{c-1}, v_3 \in R_+^{c-1}, \\ &\quad h_i = v_1,\ h_i = v_2, v_1 \in \mathcal{V}_{box},\ v_2 \in \mathcal{V}_{sph}. \end{aligned} \tag{5}$$

The augmented Lagrange function w.r.t. Eq.(5) is:

$$\begin{aligned} &L(h_i, v_1, v_2, v_3, k_1, k_2, k_3) = \sum_{j\neq i} h_i^T h_j + k_1^T (h_i - v_1) + \\ &\frac{\mu}{2}||h_i - v_1||_2^2 + k_2^T (h_i - v_2) + \frac{\mu}{2}||h_i - v_2||_2^2 \\ &+ k_3^T (h_i^T H_{\sim i} + v_3 - e) + \frac{\mu}{2}||h_i^T H_{\sim i} + v_3 - e||_2^2 \\ &s.t.\ v_1 \in V_{box}, v_2 \in V_{sph}, v_3 \in R_+^{c-1}, \end{aligned} \tag{6}$$

Algorithm 1 Optimization Procedure to Generate Hash Centers

Initialize: initialize $h_1, ..., h_c$ by Hadamard matrix and Bernoulli sampling [26]. $\rho = 1.02, \max_\mu = 10^{10}, \epsilon = 10^{-6}, T = 50$.
For t = 1,2,...,T
  For i=1,2, . . . , c
    Set $v_1, v_2, v_3, k_1, k_2, k_3$ to be zero vectors. Set $\mu = 10^{-6}$.
    Repeat
      Update h_i via Eq.(7).
      Update v_1, v_2, v_3 via Eq.(10).
      Update k_1, k_2, k_3 via Eq.(11).
      Update μ by $\mu \leftarrow \min(\rho\mu, max_\mu)$.
    Until $\max(||h_i - v_1||_\infty, ||h_i - v_2||_\infty, ||h_i^T H_{\sim i} + v_3 - e||_\infty) \leq \epsilon$.
  End For
  $T \leftarrow T + 1$.
End For
Output: h_i $(i = 1, 2, ...c)$

where $e = (q - 2d)1_{c-1}$, and k_1, k_2, k_3 are Lagrange multipliers.

Next we will present the update rules for each of the variables h_i, v_1, v_2, v_3 respectively, by minimizing L in Eq.(6) with other variables being fixed.

Update h_i By fixing other variables except h_i, the subproblem of L in Eq.(6) w.r.t. h_i is an unconstrained objective. The gradient of L w.r.t. h_i is

$$\begin{aligned}\frac{\partial L(h_i)}{\partial h_i} =& 2\mu h_i + \mu H_{\sim i} H_{\sim i}^T h_i + \sum_{j \neq i} h_j + k_1 + k_2 + H_{\sim i} k_3 \\ & - \mu(v_1 + v_2 + H_{\sim i} e - H_{\sim i} v_3).\end{aligned}$$

By setting this gradient to zeros, we can update h_i by

$$\begin{aligned}h_i \leftarrow & (2\mu I_q + \mu H_{\sim i} H_{\sim i}^T)^{-1} (\mu(v_1 + v_2 + H_{\sim i} e - H_{\sim i} v_3) \\ & - \sum_{j \neq i} h_j - k_1 - k_2 - H_{\sim i} k_3).\end{aligned} \tag{7}$$

Update v_1, v_2, v_3 The subproblem of L in Eq.(6) w.r.t. v_1, v_2, v_3 are

$$\begin{cases} L(v_1) = k_1^T (h_i - v_1) + \dfrac{\mu ||h_i - v_1||_2^2}{2} & s.t. v_1 \in V_{box} \\ L(v_2) = k_2^T (h_i - v_2) + \dfrac{\mu ||h_i - v_2||_2^2}{2} & s.t. v_2 \in V_{sph} \\ L(v_3) = k_3^T (h_i^T H_{\sim i} + v_3 - e) + \dfrac{\mu}{2} ||h_i^T H_{\sim i} + v_3 - e||_2^2 & \\ \qquad s.t. v_3 \in R_+^{c-1} & \end{cases} \tag{8}$$

By setting the gradients to zeros, we obtain the update rules for v_1, v_2 and v_3.

$$\begin{cases} v_1 \leftarrow P_{V_{box}}(h_i + \dfrac{k_1}{\mu}) \\ v_2 \leftarrow P_{V_{sph}}(h_i + \dfrac{k_2}{\mu}) \\ v_3 \leftarrow P_{R_+^{c-1}}(e - h_i^T H_{\sim i} - \dfrac{k_3}{\mu}) \end{cases} \tag{9}$$

Note that we need to project v_1, v_2, v_3 onto V_{box}, V_{sph} and R_+^{c-1}, respectively. Following [23], all of these projections have closed form solutions:

$$\begin{cases} v_1 \leftarrow \max(-1, min(1, h_i + \dfrac{k_1}{\mu})) \\ v_2 \leftarrow \sqrt{q} \dfrac{h_i + \frac{k_2}{\mu}}{||h_i + \frac{k_2}{\mu}||_2} \\ v_3 \leftarrow \max(0, e - h_i^T H_{\sim i} - \dfrac{k_3}{\mu}) \end{cases} \tag{10}$$

Update k_1, k_2, k_3 The Lagrange multipliers k_1, k_2 and k_3 can be updated by

$$\begin{cases} k_1^t = k_1^{t-1} + \mu(h_i - v_1) \\ k_2^t = k_2^{t-1} + \mu(h_i - v_2) \\ k_3^t = k_3^{t-1} + \mu({h_i}^T H_{\sim i} + v_3 - e) \end{cases} \tag{11}$$

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

3.1.3 Deriving the minimal distance d

For the objective in Eq.(1), we need to set the minimal distance parameter d so that, for any two centers h_i, h_j $(i \neq j)$, the Hamming distance $||h_i - h_j||_H$ is not less than d. We need a large d to make the hash centers to be far from each other. But d cannot be too large to ensure the feasibility of the minimal distance constraint in Eq.(1). Here we determine a large d by adopting the Gilbert-Varshamov bound [20] from coding theory.

Specifically, for c q-bit binary codes $h_i \in \{-1, 1\}^q$ $(1 \leq i \leq c)$, the Hamming distance of any two codes is at least d, the Gilbert-Varshamov bound establishes that, there exist c q-bit codes that the minimal Hamming distance of any two codes is d, as long as c, q and d satisfy

$$\frac{2^q}{c} \leq \sum_{i=0}^{d-1} \binom{q}{i} \tag{12}$$

To generate c q-bit hash centers, as long as we set a d satisfying Eq.(12), the Gilbert-Varshamov bound ensures the feasibility of the minimal distance constraint in Eq.(1). Hence, to obtain a large d, we only need the maximum of d satisfying Eq.(12). The function $f(d) = \sum_{i=0}^{d} \binom{q}{i}$ is monotonically increasing w.r.t. d. Hence, let $d^\star$ be the maximum of d satisfying Eq.(12), we have

$$\begin{cases} \frac{2^q}{c} \leq \sum_{i=0}^{d^\star - 1} \binom{q}{i} \\ \frac{2^q}{c} > \sum_{i=0}^{d^\star - 2} \binom{q}{i} \end{cases} \tag{13}$$

Since $d^\star$ is an integer in $\{1, 2, ..., q\}$, we can find it easily by exhaustive search.

Table 1. The values of the minimal distance parameter d for different hash code lengths, on three datasets used in our experiments.

dataset	16bits	32bits	64bits
ImageNet (100 classes)	4	10	24
Stanford CARs (196 classes)	4	10	23
NABirds (555 classes)	3	9	21

We set $d = d^{\star}$ as the minimal distance parameter in Eq.(1). Table 1 shows the values of the minimal distance parameter d that are derived from Eq.(13), with different code lengths on three datasets used in our experiments.

3.2. Stage 2: Train a Deep Hashing Network

With the obtained hash centers as supervised information, we train a deep hashing network which can convert input images to hash codes. As shown in Fig. 1, this deep network consists of three blocks.

The first block is a ResNet-50 [6] backbone, which consists of stacked convolution layers to capture the feature representation of an input image.

The second block is a hash code layer implemented by a fully-connected layer with TanH activations, which converts image features to an approximate hash-code vector with all elements restricted to the range $[-1, +1]$. In prediction, we use simple quantization to convert this approximate hash code to a binary code.

The third block consists of three loss functions. With the hash centers obtained from Stage 1, the first loss function is designed to make the hash code of an image to be close to the hash center of its class, but simultaneously be far from the hash centers of other classes. The second loss function is a pairwise loss that makes a pair of images in the same class have nearby hash codes. The third loss function is to reduce quantization errors. Next we will illustrate these loss functions, respectively.

3.2.1 Loss towards Hash Centers

After obtaining c hash centers, we assign one center to one of the c image classes. We develop a loss function that makes an image's output (approximate) hash code to be nearby to the center that is assigned to the class of this image, and to be faraway from other hash centers. Specifically, given c hash centers $h_1, h_2, ..., h_c$, N image $I_1, I_2, ..., I_N$ whose output hash codes are $b_1, b_2, ..., b_N$, respectively, the loss function towards hash centers is defined by

$$L_C = -\frac{1}{N}\sum_{j=1}^{N}\sum_{i=1}^{c} y_{j,i}\log P_{j,i} + (1-y_{j,i})\log(1-P_{j,i}) \tag{14}$$

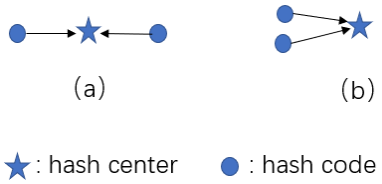


Figure 2. An illustrative example to show that, for a similar image pair whose hash codes are close towards the same hash center, the distance between their hash codes may be varying, e.g., the distance between the two hash codes in (a) is larger than that between the two hash codes in (b).

with

$$P_{j,i} = \frac{exp[-S(b_j, h_i)]}{\sum_{m=1}^{c} exp[-S(b_j, h_m)]}, \tag{15}$$

where we define $S(x, y)$ as a metric of similarities between x and y, $y_{j,i}$ is an indicator that $y_{j,i} = 1$ if the image I_j belongs to the i-th class whose hash center is h_i, otherwise $y_{j,i} = 0$. Following the existing hashing methods [4, 7], we use the scaled cosine similarity as the similarity metric, so $P_{j,i}$ in Eq.(15) can be reformulated as:

$$P_{j,i} = \frac{exp[\sqrt{q}\cos(b_j, h_i)]}{\sum_{m=1}^{c} exp[\sqrt{q}\cos(b_j, h_m)]}, \tag{16}$$

where we define $\cos(x, y) = \frac{x^T y}{||x||_2||y||_2}$ as the cosine similarities between x and y, q is the length of hash codes.

3.2.2 Loss for Similar Pairs of the Same Hash Center

Consider images I_x and I_y whose hash codes are b_x and b_y respectively, I_x and I_y belong to the same class that is assigned a hash center h. The loss function towards hash centers in Eq.(14) makes that b_x and b_y to be close to the hash center h. Suppose both b_x and b_y are close to h, however, the distance between b_x and b_y may be diverse depending on their relative position. Fig.2 shows an illustrative example. In both Fig.2(a) and Fig.2(b), b_x and b_y have the same small distance to h. However, the distance between b_x and b_y in Fig.2(a) is much larger than that in Fig.2(b).

Inspired by the above observation, we develop a loss to make the Hamming distance of the similar images' hash codes to be small. This loss is defined by

$$L_P = -\sum_{I_x, I_y \ are \ similar} \log\frac{1}{1+e^{D(b_x, b_y)}} \tag{17}$$

where $D(x, y)$ is the distance metric of x and y. Eq.(17) uses the negative log likelihood function to make the distance $D(b_x, b_y)$ as small as possible.

Note that for two binary codes $h_1, h_2 \in \{-1, 1\}^q$, we have the scaled Hamming distance $\frac{1}{q}||h_1 - h_2||_H =$

Table 2. Comparison results of retrieval performance w.r.t. MAP on three datasets

Methods	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
DSH [13]	0.7179	0.7448	0.7585	0.0820	0.1011	0.2030	0.2153	0.3124	0.4309
DPSH [11]	0.6241	0.7626	0.7992	0.1171	0.1855	0.2811	0.1764	0.2949	0.4132
HashNet [2]	0.6024	0.7158	0.8071	0.0825	0.1439	0.2359	0.2637	0.3611	0.4845
DTSH [22]	0.6606	0.7580	0.8120	0.1307	0.1410	0.2218	0.4251	0.5461	0.6553
GreedyHash [18]	0.7394	0.7977	0.8243	0.3519	0.5350	0.6177	0.7312	0.8271	0.8432
DPN [3]	0.7987	0.8298	0.8394	0.6151	0.6928	0.7244	0.7287	0.8214	0.8488
CSQ [26]	0.8377	0.8750	0.8836	0.6183	0.7210	0.7491	0.7435	0.8392	0.8634
OrthoHash [7]	0.8540	0.8792	0.8936	0.6366	0.7243	0.7544	0.8012	0.8490	0.8676
Ours	**0.8639**	**0.8863**	**0.9019**	**0.6977**	**0.7417**	**0.7619**	**0.8579**	**0.8731**	**0.8814**

$\frac{q-h_1^T h_2}{2q}$. Since b_i and b_j are continuous vectors that cannot be used in calculating Hamming distance, we set $D(b_x, b_y) = \frac{q-b_i^T b_j}{2q}$ as an approximation of the scaled Hamming distance. Hence Eq.(17) can be simplified as

$$L_P = \sum_{I_x, I_y \ are \ similar} \log(1 + e^{\frac{q-b_x^T b_y}{2q}}) \tag{18}$$

3.2.3 Quantization loss

In the proposed network, the output hash code is continuous because the hash code layer is implemented by a fully connected layer with tanh activations. To reduce quantization errors, similarly to the existing methods (e.g, [27]), we use a loss with bi-modal Laplacian prior, which is defined as:

$$L_Q = \sum_{i=1}^{N} ||||b_i||_1 - \mathbf{1}||_1, \tag{19}$$

where $||.||_1$ is the ℓ_1 norm, $\mathbf{1}$ is the vector with all ones.

3.2.4 Combination of Loss Functions

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

$$L = L_C + \lambda_1 L_P + \lambda_2 L_Q \tag{20}$$

where λ_1, λ_2 are trade-off hyper-parameters.

4. Experiments

4.1. Experiment Settings

We conduct experiments on three datasets for image retrieval, ImageNet [17], NABirds [19] and Stanford Cars [8]. Each dataset has hundred(s) of image classes. Following the settings in [2, 26], the ImageNet dataset contains 143,495 images in 100 classes, where we use 10,000 images for training (100 images from each of the 100 classes), 5,000 query images for test, and the remains as the retrieval database. The Stanford Cars dataset contains 16,185 images in 196 classes. The NABirds dataset contains 48,562 images in 555 classes. On NABirds/Stanford Cars, we use the official train set as the training images and the retrieval database, the official test set as the test queries for retrieval.

To evaluate the retrieval performance, we use two widely-used metrics: Mean Average Precision (MAP) and Precision-Recall curves. Since ImageNet has a large-size retrieval database, following [26], we use MAP@1000 as the MAP metric on ImageNet. On NABirds and Stanford Cars, we use MAP@ALL as the MAP metric.

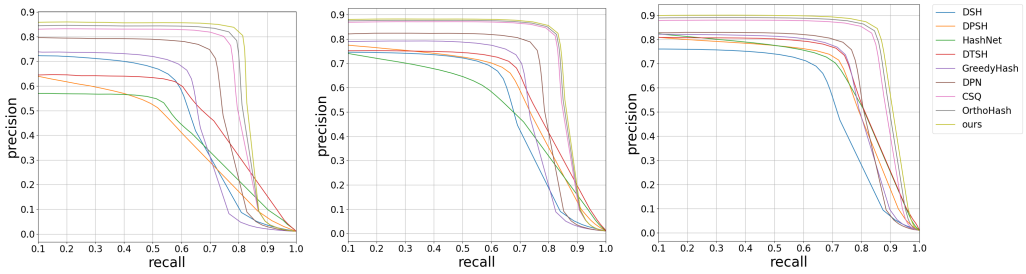
We compare the performance of the proposed method with eight deep hashing baselines, including three state-of-the-art pointwise methods that use hash centers (OrthoHash [7], CSQ [26], DPN [3]), and five other deep hashing methods (Greedy Hash [18], DTSH [22], HashNet [2], DPSH [11], DSH [13]). For fair comparison, all of these methods use the ResNet50 pre-trained model as backbones.

4.2. Implementation Details

The proposed deep network is implemented with Pytorch [16] on a server with GeForce RTX 3090 Ti GPUs. In all experiments, our deep hashing network are optimized by RMSProp and the mini-batch size of images is 64. For each dataset, we randomly split the training images into to a validation set with 20% images and a training set with the rest 80% images. Then we train models with the training set and determine λ_1, λ_2 used in Eq.20 by the validation set. Finally, we train the final models by using all of the training images and the obtained λ_1 and λ_2. After training the deep hashing network, the hash code of an image is computed by $sign(b)$, where b is the image's output approximate hash code of the deep hashing network.

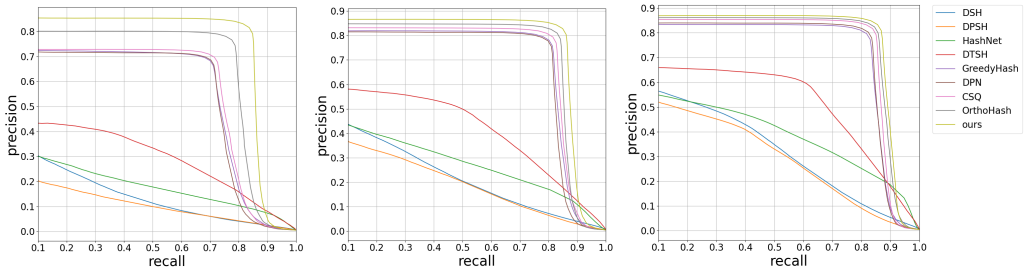
Table 3. Comparison results w.r.t. MAP for different combinations of loss functions

L_C	L_P	L_Q	ImageNet 16bits	ImageNet 32bits	ImageNet 64bits	Stanford CAR 16bits	Stanford CAR 32bits	Stanford CAR 64bits	NABirds 16bits	NABirds 32bits	NABirds 64bits
✓	✓	✓	0.8639	0.8863	0.9019	0.8579	0.8731	0.8814	0.6977	0.7417	0.7619
✓		✓	0.8561	0.8852	0.8984	0.8221	0.8587	0.8742	0.6810	0.7299	0.7552
✓	✓		0.8601	0.8858	0.8995	0.8451	0.8672	0.8752	0.6937	0.7372	0.7573
✓			0.8558	0.8839	0.8977	0.8182	0.8578	0.8750	0.6715	0.7266	0.7548
CSQ-L_c			0.8457	0.8830	0.8922	0.7888	0.8467	0.8677	0.6351	0.7050	0.7480



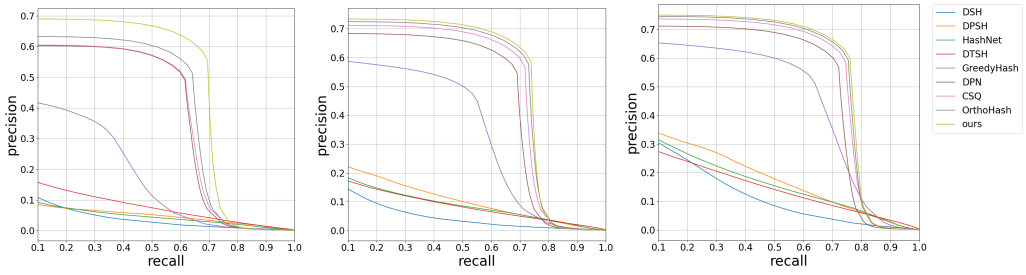
(a) 16bits (b) 32bits (c) 64bits

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



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

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



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

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

4.3. Results of Retrieval Accuracies

Table 2 displays the Mean Average Precision (MAP) for retrieval performance on three datasets. The proposed method outperforms the baseline methods on all of these datasets. To illustrate, the MAP results of the proposed method demonstrate a relative increase of **1.0**% $\sim$ **9.6**% **/ 1.6**% $\sim$ **7.1**% **/ 0.9**% $\sim$**1.2**% on NABirds / Stanford Cars / ImageNet, respectively, when compared to the best baseline. Fig.3, Fig.4 and Fig.5 show the retrieval performance w.r.t. Precision-Recall curves on three datasets, in which we can see that the proposed method outperforms all the baselines on three datasets. The proposed approach yields significant improvements in scenarios with a high number of image classes and short hash codes. To illustrate, the pro-

Table 4. Comparison results of mean Hamming distance over all pairs of hash centers and minimum Hamming distance between any two hash centers, for different ways to generate hash centers

Datasets	Methods	16bits		32bits		64bits	
		min	mean	min	mean	min	mean
ImageNet	baseline	2	8.04	6	16.11	32	32.23
	ours	4	8.08	10	16.16	32	32.23
Stanford Cars	baseline	0	8.01	6	16.03	16	32.10
	ours	4	8.05	10	16.09	23	32.16
NABirds	baseline	0	7.98	4	15.98	14	32.01
	ours	3	8.01	9	16.03	21	32.06

Table 5. Comparison results w.r.t. MAP for different ways to generate hash centers

datasets	methods	16bits	32bit	64bits
ImageNet	baseline	0.8535	0.8844	0.9019
	ours	0.8639	0.8863	0.9019
Stanford Cars	baseline	0.7977	0.8665	0.8759
	ours	0.8579	0.873	0.8814
NABirds	baseline	0.6544	0.7321	0.7545
	ours	0.6977	0.7417	0.7619

posed method displays a relative improvement of **9.6**% and **7.1**% compared to the best baseline for NABirds with 555 classes and Stanford Cars with 196 classes, respectively, when 16-bit codes are used.

4.4. Ablation Studies

4.4.1 Effectiveness of Hash Centers

Our primary contribution is the proposed optimization procedure to produce hash centers mutually separated by a worst-case guaranteed minimal distance, being avoid of badly separated hash centers with arbitrary small distance. To assess the effectiveness of these centers, a two-stage baseline is implemented. In Stage 1, hash centers are produced by Hadamand matrix and Bernoulli sampling (as in [26]). In Stage 2, a deep hashing network was trained with these centers using the same settings as the proposed method. The sole distinction between the baseline and the proposed method is the method of producing hash centers.

Table 4 presents the comparison results of the averaged Hamming distance across all pairs of hash centers and the minimum Hamming distance between any two hash centers, for the hash centers generated by the baseline or the proposed method, respectively. Three observations can be made from Table 4. (1) In all cases, the minimal distances obtained by the proposed method meet the derived values of d that are set before optimization (see Table 1), indicating the empirical success of our optimization approach. (2) In most cases, the minimal distances of the baseline are smaller than those of the proposed method, indicating some hash centers generated by the baseline are not separated well, e.g., 0 distances with 16-bit codes on NABirds/Stanford Cars. (3) In both methods, the mean distances over all hash center pairs are similar and close to a half of code lengths.

Table 5 demonstrates a clear superiority of the proposed method over the baseline on all datasets. These results, in conjunction with Table 4, validate that our optimization approach for obtaining hash centers with a guaranteed minimal distance, even under worst-case scenarios, can enhance the retrieval performance. For example, in Table 4, for 16-bit codes on NABirds and Stanford Cars, some hash centers generated by the baseline are not separated well because the minimal distances are 0, while the hash centers generated by the proposed method are separated by a minimal distance 4 (Stanford Cars) and 3 (NABirds). Simultaneously we observe that, in Table 5, the proposed method makes large improvements over the baseline with 16-bit codes, i.e., **6.6**% on NABrids and **7.5**% on Stanford Cars.

4.4.2 Effectiveness of Loss Functions

To investigate the individual improvements made by the loss functions, in the proposed deep network, we assess several combinations of the loss L_C towards hash centers, L_P for similar image pairs and L_Q for quantization.

Table 3 presents the MAP results for four methods, each utilizing identical hash centers and network architecture. CSQ-L_c utilizes the loss L_c for hash centers proposed in CSQ [26]. The rest methods use different combinations of the losses L_C, L_P and L_Q of the proposed method. Table 3 reveals two observations: (1) The method with the proposed loss L_C outperforms the baseline CSQ-L_c, which uses a related loss for hash centers, thereby validating the effectiveness of L_C. (2) The loss L_P which is designed for similar image pairs, also enhances the retrieval performance.

5. Conclusions

This paper presents an optimization approach for producing hash centers mutually separated with a guaranteed minimal distance, using the Gilbert-Varshamov bound to obtain a large distance while maintaining optimization feasibility. With these centers, we propose effective loss functions to train deep hashing networks. Empirical evaluations for image retrieval show that the proposed method outperforms the state-of-the-art methods.

Acknowledgements This work was supported by Guangdong Basic and Applied Basic Research Foundation (2023A1515011400, 2019B1515130001, 2021A1515012172), National Science Foundation of China (61772567, U1811262).

References

[1] Yue Cao, Mingsheng Long, Bin Liu, and Jianmin Wang. Deep Cauchy Hashing for Hamming Space Retrieval. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-18)*, pages 1229–1237, 2018. 1, 2

[2] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Philip S Yu. HashNet: Deep Learning to Hash by Continuation. *Proceedings of the IEEE International Conference on Computer Vision (ICCV-17)*, pages 5608–5617, 2017. 2, 6

[3] Lixin Fan, Kam Woh Ng, Ce Ju, Tianyu Zhang, and Chee Seng Chan. 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)*, pages 825–831, 2020. 2, 6

[4] Yunchao Gong, Sanjiv Kumar, Vishal Verma, and Svetlana Lazebnik. Angular Quantization-based Binary Codes for Fast Similarity Search. *Advances in Neural Information Processing Systems (NIPS-12)*, pages 1196–1204, 2012. 5

[5] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. 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. 2

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-16)*, pages 770–778, 2016. 5

[7] Jiun Tian Hoe, Kam Woh Ng, Tianyu Zhang, Chee Seng Chan, Yi-Zhe Song, and Tao Xiang. One Loss for All: Deep Hashing with a Single Cosine Similarity based Learning Objective. *Advances in Neural Information Processing Systems (NIPS-21)*, 2021. 2, 5, 6

[8] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3D Object Representations for Fine-Grained Categorization. *Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCV-13)*, pages 554–561, 2013. 6

[9] Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan. Simultaneous Feature Learning and Hash Coding With Deep Neural Networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-15)*, pages 3270–3278, 2015. 2

[10] Qi Li, Zhenan Sun, Ran He, and Tieniu Tan. Deep Supervised Discrete Hashing. *Advances in Neural Information Processing Systems (NIPS-17)*, pages 2479–2488, 2017. 1, 2

[11] Wujun Li, Sheng Wang, and Wangcheng Kang. Feature Learning Based Deep Supervised Hashing with Pairwise Labels. *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)*, pages 1711–1717, 2016. 2, 6

[12] Kevin Lin, Huei-Fang Yang, Jen-Hao Hsiao, and Chu-Song Chen. Deep Learning of Binary Hash Codes for Fast Image Retrieval. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 27–35, 2015. 2

[13] Haomiao Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen. Deep Supervised Hashing for Fast Image Retrieval. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-16)*, pages 2064–2072, 2016. 2, 6

[14] Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. Supervised hashing with kernels. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-12)*, pages 2074–2081, 2012. 2

[15] Wei Liu, Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Hashing with graphs. *Proceedings of the 28th International Conference on International Conference on Machine Learning (ICML-11)*, pages 1–8, 2011. 2

[16] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems (NIPS-19)*, page 8024–8035. Curran Associates, Inc., 2019. 6

[17] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 6

[18] Shupeng Su, Chao Zhang, Kai Han, and Yonghong Tian. Greedy Hash: Towards Fast Optimization for Accurate Hash Coding in CNN. *Advances in Neural Information Processing Systems (NIPS-18)*, pages 806–815, 2018. 1, 2, 6

[19] Grant Van Horn, Steve Branson, Ryan Farrell, Scott Haber, Jessie Barry, Panos Ipeirotis, Pietro Perona, and Serge Belongie. 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)*, pages 595–604, 2015. 6

[20] Rom Rubenovich Varshamov. Estimate of the number of signals in error correcting codes. *Docklady Akad. Nauk, SSSR*, 117:739–741, 1957. 1, 3, 4

[21] J. Wang, T. Zhang, J. Song, N. Sebe, and H. Shen. A survey on learning to hash. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(04):769–790, 2018. 2

[22] Xiaofang Wang, Yi Shi, and Kris M Kitani. Deep Supervised Hashing with Triplet Labels. *Asian Conference on Computer Vision (ACCV-16)*, pages 70–84, 2016. 1, 2, 6

[23] Baoyuan Wu and Bernard Ghanem. ℓ_p-Box ADMM: A Versatile Framework for Integer Programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(7):1695–1708, 2019. 1, 3, 4

[24] Rongkai Xia, Yan Pan, Hanjiang Lai, Cong Liu, and Shuicheng Yan. Supervised Hashing for Image Retrieval via Image Representation Learning. *Proceedings of the AAAI conference on Artificial Intelligence (AAAI-14)*, pages 2156–2162, 2014. 1, 2

[25] Huei-Fang Yang, Kevin Lin, and Chu-Song Chen. Supervised Learning of Semantics-Preserving Hash via Deep Con-

volutional Neural Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(2):437–451, 2017. 2

[26] Li Yuan, Tao Wang, Xiaopeng Zhang, Francis EH Tay, Zequn Jie, Wei Liu, and Jiashi. Feng. Central Similarity Quantization for Efficient Image and Video Retrieval. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-20)*, pages 3083–3092, 2020. 1, 2, 4, 6, 8

[27] Han Zhu, Mingsheng Long, Jianmin Wang, and Yue Cao. Deep Hashing Network for Efficient Similarity Retrieval. *Proceedings of the AAAI conference on Artificial Intelligence (AAAI-16)*, pages 2415–2421, 2016. 6