Codebook-Centric Deep Hashing: End-to-End Joint Learning of Semantic Hash Centers and Neural Hash Function

Shuo Yin[1], Zhiyuan Yin[1], Yuqing Hou[2], Rui Liu[3], Yong Chen[1]*, Dell Zhang[4]
[1]School of Computer Science, Beijing University of Posts and Telecommunications, Beijing, China
[2]Independent Researcher
[3]School of Computer Science and Engineering, Beihang University, Beijing, China
[4]Institute of Artificial Intelligence (TeleAI), China Telecom, Shanghai, China
yinshuo06@163.com, zy_yin@bupt.edu.cn, houyuqing1988@gmail.com, lr@buaa.edu.cn, dell.z@ieee.org

Abstract

Hash center-based deep hashing methods improve upon pairwise or triplet-based approaches by assigning fixed hash centers to each class as learning targets, thereby avoiding the inefficiency of local similarity optimization. However, random center initialization often disregards inter-class semantic relationships. While existing two-stage methods mitigate this by first refining hash centers with semantics and then training the hash function, they introduce additional complexity, computational overhead, and suboptimal performance due to stage-wise discrepancies. To address these limitations, we propose **Center-Reassigned Hashing (CRH)**, an end-to-end framework that **dynamically reassigns hash centers** from a preset codebook while jointly optimizing the hash function. Unlike previous methods, CRH adapts hash centers to the data distribution **without explicit center optimization phases**, enabling seamless integration of semantic relationships into the learning process. Furthermore, **a multi-head mechanism** enhances the representational capacity of hash centers, capturing richer semantic structures. Extensive experiments on three benchmarks demonstrate that CRH learns semantically meaningful hash centers and outperforms state-of-the-art deep hashing methods in retrieval tasks.

Code — https://github.com/iFamilyi/CRH
Extended version — https://arxiv.org/abs/2511.12162

Introduction

Image hashing has become a cornerstone of large-scale image retrieval systems due to its remarkable computational efficiency and compact storage footprint. The central objective is to map high-dimensional image data into low-dimensional binary codes that preserve semantic similarity in Hamming space. With the advent of deep learning, deep hashing has emerged as the dominant paradigm, significantly surpassing traditional shallow methods by learning powerful deep neural hash functions. This work focuses on the deep supervised hashing, where label information explicitly guides the learning of discriminative binary codes.

Existing deep supervised hashing methods can be broadly categorized by their similarity modeling strategies into three groups: **pairwise**, **triplet**, and **pointwise** approaches. Pairwise (Li, Wang, and Kang 2016; Cao et al. 2017) and triplet (Lai et al. 2015; Wang, Shi, and Kitani 2016) methods aim to preserve local similarity relationships among pairs or triplets of samples, often incurring quadratic or higher computational complexity with respect to the number of samples. In contrast, pointwise methods (Yang, Lin, and Chen 2018; Su et al. 2018; Yuan et al. 2020) directly leverage class labels, achieving linear complexity.

Recently, pointwise approaches based on **hash centers** have attracted increasing attention (Yuan et al. 2020; Hoe et al. 2021; Wang et al. 2022, 2023; Xiao et al. 2024). These methods predefine a binary hash center for each class and train networks to align image representations with their assigned centers. While state-of-the-art models such as CSQ (Yuan et al. 2020), OrthoHash (Hoe et al. 2021), and MDS (Wang et al. 2023) have achieved impressive results, they typically fix the assignment of hash centers to classes randomly at initialization. This overlooks inter-class semantic correlations—e.g., semantically similar classes like “cat” and “dog” should ideally have closer hash centers than unrelated classes like “cat” and “car”—limiting their ability to capture global semantic structures.

To mitigate this, SHC (Chen et al. 2025) adopts a two-stage strategy, first generating semantically aware hash centers via classifier-based similarity estimation and iterative optimization, then learning the hash function. Although effective, this approach introduces heavy computational overhead, breaks end-to-end trainability, and relies on similarity measures rooted in classification objectives, which may not align perfectly with retrieval goals.

To address these issues, we propose **Center-Reassigned Hashing (CRH)**, a novel framework that dynamically optimizes hash center assignments during hash function training. Unlike existing methods that fix center assignments after initialization, CRH iteratively reassigns centers throughout training, progressively aligning them with inter-class semantic relationships. Concretely, CRH comprises three key components: (1) hash codebook initialization, followed by alternating stages of (2) hash function optimization and (3) hash center reassignment. This design eliminates the need for explicit pre-training or offline hash center generation, enabling direct end-to-end learning. Crucially, our “reassignment” refers to refining class-to-center assignments, not al-

---

*Corresponding author (alphawolf.chen@gmail.com).

tering the binary codebook itself.

To sum up, our contributions are as follows:

- We introduce a novel *reassignment-based mechanism* for updating hash centers, enabling the joint learning of hash function and semantic hash centers without requiring separate pre-training or extra hash center generation phase.
- We propose a flexible *hash codebook* consisting of $M(\geq C)$ candidate hash centers, along with *a multi-head design* (e.g., H-head) that effectively expands the codebook capacity from M to M^H, allowing finer-grained semantic representations without increasing the physical codebook size M.
- Rich experiments on single-label and multi-label benchmarks demonstrate that our CRH method can effectively learns semantically meaningful hash centers and achieve consistent superiority over existing state-of-the-art deep hashing methods.

Related Work

Hashing approaches can be broadly categorized into data-independent (Indyk and Motwani 1998; Gionis, Indyk, and Motwani 1999; Charikar 2002) and data-dependent methods (Weiss, Torralba, and Fergus 2008; Kulis and Darrell 2009; Norouzi and Fleet 2011; Kong and Li 2012; Liu et al. 2012; Gong et al. 2013). The latter, particularly deep hashing methods (Li, Wang, and Kang 2016; Li et al. 2017; Jiang and Li 2017, 2018; Hoe et al. 2021; Lu et al. 2022; Wang et al. 2022, 2023), have become dominant due to their ability to learn compact and discriminative hash codes. Deep hashing techniques can be further classified into pairwise, triplet, and pointwise methods based on their supervision paradigms.

Pairwise and Triplet Methods learn hash functions by preserving local similarity structures. Pairwise approaches (Li, Wang, and Kang 2016; Cao et al. 2017, 2018) minimize/maximize distances between similar/dissimilar pairs, while triplet methods (Lai et al. 2015; Wang, Shi, and Kitani 2016) enforce relative distance constraints. Although effective, these methods suffer from $\mathcal{O}(N^2)$ (N is the number of data samples) or more computational complexity and struggle to capture global data structures (Yuan et al. 2020).

Pointwise Methods overcome these limitations by utilizing direct label supervision. Early approaches (Yang, Lin, and Chen 2018; Su et al. 2018) treated hashing as a classification problem. More recent center-based methods (Yuan et al. 2020; Fan et al. 2020; Hoe et al. 2021; Wang et al. 2023) assign predefined hash centers to each class, achieving $\mathcal{O}(NC)$ (C denotes the number of classes) complexity and better global similarity preservation. However, these approaches typically generate centers through random sampling (CSQ) or combinatorial optimizations (MDS), ignoring inter-class semantic relationships.

Recent work (Wang et al. 2022; Chen et al. 2025) proposed a two-stage method that first injects semantics into hash centers via class relationship estimation and discrete optimization, then learns hash function with these fixed centers. While effective, these approaches require solving NP-hard optimization problems and suffer from stage-wise optimization gaps. In contrast, our CRH framework dynamically adjusts hash centers during end-to-end training, naturally encoding semantic relationships.

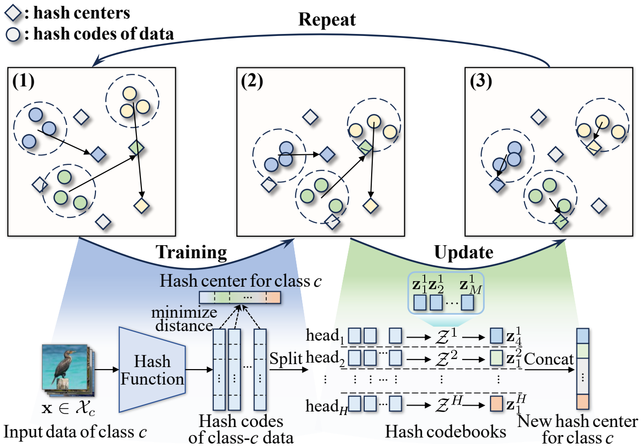


Figure 1: The overall framework of CRH. **Top**: Hamming space visualization of the iterative hash center reassignment across 3 stages: (1) initial or previous assignment, (2) hash code convergence, and (3) updated center assignment. Three colors represent three classes. **Bottom**: hash function training and multi-head update process for class c, where each head independently updates its sub-center $\mathbf{z}_m^h$ on a hash split, followed by concatenation into the full center $\mathbf{c}_c$.

The Proposed Method

Problem Statement

Given a dataset $\mathcal{X} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$ with N samples, each feature vector $\mathbf{x}_n \in \mathbb{R}^D$ is associated with a multi-hot label vector $\mathbf{y}_n = [y_{n1}, \cdots, y_{nC}]^\top \in \{0,1\}^C$. Here, C denotes the total number of classes, and $y_{nc} = 1$ indicates that the n-th sample belongs to the c-th class, while $y_{nc} = 0$ otherwise. The objective of supervised hashing is to learn an encoding function $\mathrm{f} : \mathbb{R}^D \to \{-1,1\}^K$ that maps an input $\mathbf{x}$ to a compact K-bit binary hash code $\mathrm{f}(\mathbf{x})$. This function should preserve semantic similarity, ensuring that the Hamming distance between the hash codes of samples sharing at least one common category is smaller than that of samples with no overlapping categories.

Overall Framework

Figure 1 illustrates the overall framework of CRH. (1) We begin by randomly constructing a codebook $\mathcal{Z}$ comprising M binary vectors of length K, where M exceeds the number of classes C. We then randomly select C vectors from $\mathcal{Z}$ to serve as the initial hash centers for the C classes. (2) Given these class-specific hash centers, we train a hash function that maps input samples to their binary hash codes. (3) After training, we update the class hash centers by reassigning them using all training samples' current hash codes and the codebook: specifically, we identify C vectors from the codebook $\mathcal{Z}$ that minimize the total distance between the training samples' hash codes and their nearest class centers, effectively yielding a new set of hash centers. (4) Steps (2) and (3) are iteratively repeated until convergence, enabling the simultaneous learning of semantically structured hash centers and the hash function. (5) Notably, inspired by multi-head attention, during the center update process we partition the codebook $\mathcal{Z}$ into H sub-codebooks $\{\mathcal{Z}^h\}_{h=1}^H$. Each sub-codebook independently produces a sub-hash center $\mathbf{z}_m^h$ ($m \in \{1, \cdots, M\}$, $h \in \{1, \cdots, H\}$), and the final hash center $\mathbf{c}_c$ ($c \in \{1, \cdots, C\}$) for each class is formed by concatenating these sub-hash centers.

The detailed procedures for steps (1), (2), and (3) are described as follows.

Initialization of Hash Centers

In contrast to existing hash-center-based methods that assign one hash center per category (i.e., resulting in exactly C hash centers for a C-class dataset), we propose to construct an expanded pool of candidate hash centers. Specifically, we define a *hash codebook* $\mathcal{Z} = \{\mathbf{z}_m\}_{m=1}^M$ consisting of M binary hash codes $\mathbf{z}_m \in \{-1, 1\}^K$ ($M \geq C$). This codebook serves as a candidate set from which class-specific hash centers are selected—each class center must be assigned a distinct element from $\mathcal{Z}$.

$\mathcal{Z}$ is initially populated by sampling from the Bernoulli distribution Bern(0.5) (Yuan et al. 2020). In this approach, each bit of every hash code is independently set to -1 or 1 with equal probability. However, when K is small (e.g., $K \leq 16$) and M is large, this independent sampling may lead to duplicate hash codes within $\mathcal{Z}$, reducing the effective diversity of the codebook.

To address this problem, we instead uniformly sample M unique binary vectors from the full space $\{-1, 1\}^K$ to construct $\mathcal{Z}$ for small K, of which the procedure is computationally efficient. Moreover, under this uniform sampling strategy, the expected average Hamming distance between any two hash codes in $\mathcal{Z}$ is at least $K/2$ (Yuan et al. 2020), ensuring sufficient separation among the candidate centers.

Once the hash codebook $\mathcal{Z}$ is constructed, we randomly select C distinct elements from it to serve as the initial class-specific hash centers $\{\mathbf{c}_c\}_{c=1}^C$, where $\mathbf{c}_c$ denotes the hash center assigned to the c-th class. These assignments will be dynamically refined during training through our proposed center reassignment mechanism.

Training of Hash Function

Given the initialized or updated hash centers, we train the hash function $\mathrm{f}(\mathbf{x})$, implemented as a deep neural network (e.g., a pre-trained ResNet-34 backbone followed by a linear projection layer with a $\tanh$ activation), to map each input sample $\mathbf{x}_n$ to a continuous binary-like code $\mathbf{h}_n = \tanh(\mathbf{v}_n) \in (-1, 1)^K$, where $\mathbf{v}_n$ denotes the output of the linear layer. For retrieval, these codes are subsequently binarized as $\mathbf{b}_n = \mathrm{sign}(\mathbf{h}_n)$.

We adopt a margin-based cross-entropy loss $\mathcal{L}_{\mathrm{CE}}$ (Hoe et al. 2021), defined as:

$$\mathcal{L}_{\mathrm{CE}} = -\frac{1}{N} \sum_{n=1}^{N} \sum_{c=1}^{C} \frac{y_{nc}}{\|\mathbf{y}_n\|_1} \log p_{nc}, \tag{1}$$

with softmax probability:

$$p_{nc} = \frac{\exp(s \cdot \mathrm{sim}(\mathbf{v}_n, \mathbf{c}_c))}{\sum_{j=1}^{C} \exp(s \cdot \mathrm{sim}(\mathbf{v}_n, \mathbf{c}_j))}, \tag{2}$$

where $\mathrm{sim}(\mathbf{v}_n, \mathbf{c}_c) = \mathbf{v}_n^\top \mathbf{c}_c / (\|\mathbf{v}_n\|_2 \cdot \|\mathbf{c}_c\|_2) - y_{nc} \cdot margin$ incorporates cosine similarity with a margin $margin$, s is a scaling factor, and $\|\mathbf{y}_n\|_1 = \sum_{c=1}^{C} |y_{nc}|$ counts the number of labels for the n-th data point. This loss pulls embeddings toward their corresponding hash centers while pushing them away from others.

To minimize quantization errors due to relaxations in the $\tanh$ layer, we introduce a quantization loss:

$$\mathcal{L}_q = \frac{1}{NK} \sum_{n=1}^{N} \|\mathrm{abs}(\mathbf{h}_n) - \mathbf{1}\|_2^2 = \frac{1}{NK} \sum_{n=1}^{N} \sum_{k=1}^{K} (|h_{nk}| - 1)^2, \tag{3}$$

where $\mathrm{abs}(\cdot)$ denotes the element-wise absolute value, $\mathbf{1}$ is a vector of all ones, and $\|\cdot\|_2$ is the ℓ_2-norm.

The overall objective is:

$$\mathcal{L} = \mathcal{L}_{\mathrm{CE}} + \lambda \mathcal{L}_q, \tag{4}$$

where $\lambda (\geq 0)$ balances $\mathcal{L}_{\mathrm{CE}}$ and $\mathcal{L}_q$.

Updating of Hash Centers via Reassignment

The initial assignment of hash centers to classes is random and therefore semantically agnostic. To enhance semantic alignment, we dynamically reassign hash centers throughout training by leveraging evolving data representations without directly numerically optimizing the centers themselves. The proposed reassignment procedure is detailed below.

Reassignment Strategy. At the end of selected training epochs (e.g., for every five epochs), we update the hash centers by first passing the training data through the hash function $\mathrm{f}(\mathbf{x})$ to produce continuous representations $\mathbf{h}_\mathbf{x}$, which are subsequently binarized to $\mathrm{sign}(\mathbf{h}_\mathbf{x})$. For each class c, let $\mathcal{X}_c$ denote its collection of data samples. We compute the mean Euclidean distance between $\mathrm{sign}(\mathbf{h}_\mathbf{x})$ and each candidate center $\mathbf{z}_m \in \mathcal{Z}$ as the error of assigning $\mathbf{z}_m$ to class c, yielding a cost matrix $\mathbf{L} = (l_{cm})_{C \times M}$ defined by:

$$l_{cm} = \frac{1}{|\mathcal{X}_c|} \sum_{\mathbf{x} \in \mathcal{X}_c} \|\mathrm{sign}(\mathbf{h}_\mathbf{x}) - \mathbf{z}_m\|_2^2, \tag{5}$$

where $c = 1, 2, \cdots, C$, $m = 1, 2, \cdots, M$, and $|\mathcal{X}_c|$ denotes the number of data samples in category c.

We then seek an optimal one-to-one mapping from classes to codebook elements that minimizes the total assignment error:

$$(j_1^*, j_2^*, \cdots, j_C^*) = \mathop{\arg\min}_{\substack{j_1, \cdots, j_C \in \{1, \cdots, M\} \\ j_i \neq j_k \text{ for } i \neq k}} \sum_{c=1}^{C} l_{c j_c}, \tag{6}$$

where j_c^* represents the index of the center in $\mathcal{Z}$ assigned to the c-th category in the optimal solution. This combinatorial optimization problem can be efficiently solved using the Hungarian algorithm (Kuhn 1955).

While the Hungarian algorithm provides an optimal assignment within each epoch, empirical results reveal that a *greedy algorithm* performs better overall. It sequentially assigns each class to its best unassigned center in $\mathcal{Z}$, following a random class order. This stochasticity may lead to suboptimal per-epoch assignments but helps prevent overfitting to transient local minima that shift during training.

Multi-Label Handling. In multi-label settings, the supervision for each sample reflects a mixture of categories, diluting the representativeness for any single class. To address this, we follow DCSH (Jose et al. 2022) by weighting each data point's contribution with factor $1/\|\mathbf{y}_{\mathbf{x}}\|_1$, where $\|\mathbf{y}_{\mathbf{x}}\|_1$ denotes the number of labels associated with sample $\mathbf{x}$. The weighted error for class c is:

$$l_{cm} = \frac{1}{\sum_{\mathbf{x}\in\mathcal{X}_c} \frac{1}{\|\mathbf{y}_{\mathbf{x}}\|_1}} \sum_{\mathbf{x}\in\mathcal{X}_c} \frac{1}{\|\mathbf{y}_{\mathbf{x}}\|_1} \|\text{sign}(\mathbf{h}_{\mathbf{x}}) - \mathbf{z}_m\|_2^2 . \tag{7}$$

By incorporating these weights, data points with more category labels contribute less to the computation of l_{cm}, ensuring that the cost matrix $\mathbf{L}$ more accurately reflects the distance relationships between the data of different categories and the elements in $\mathcal{Z}$. For single-label datasets, Eq. (7) naturally reduces to Eq. (5).

Multi-Head Codebook Design. To amplify the representational power of the hash codebook $\mathcal{Z}$ without increasing its cardinality M, we split each K-dimensional vector $\mathbf{z}_m$ into H heads (i.e., $\{\mathbf{z}_m^1, \cdots, \mathbf{z}_m^H\}$), with each $\mathbf{z}_m^h$ sized by $d = K/H$. Each head independently performs center reassignment using its sub-codebook $\mathcal{Z}^h = \{\mathbf{z}_1^h, \cdots, \mathbf{z}_M^h\}$, derived from the original codebook $\mathcal{Z}$. Specifically, for head h, the corresponding components $\mathbf{h}_{\mathbf{x}}^h$ and sub-codebook $\mathcal{Z}^h$ are used to compute a per-head cost matrix $\mathbf{L}^h$, followed by greedy assignment to yield updated centers $\{\mathbf{c}_c^h\}_{c=1}^C$. Final centers are obtained by concatenating the head-specific centers:

$$\mathbf{c}_c = \text{concat}(\mathbf{c}_c^1, \cdots, \mathbf{c}_c^H). \tag{8}$$

This multi-head design effectively enlarges the codebook capacity from M to M^H, enabling richer semantic discrimination. Since d bits can represent at most 2^d distinct binary vectors, we constrain $H \leq K/\log_2 M$ to ensure the M elements within each $\mathcal{Z}^h$ remain mutually distinct and avoid collisions. Algorithm 1 summarizes the overall procedure.

Optimization Perspective of CRH

From an optimization standpoint, center-based hashing inherently poses an NP-hard problem: determining both a C-element subset from $\{-1, 1\}^K$ and its assignment to categories that most effectively encapsulates inter-class semantic relationships, thereby maximizing retrieval performance. Our CRH method offers an efficient approximation by introducing two pivotal mechanisms: (1) **Codebook Constraint**: Instead of searching the entire Hamming space, we restrict the selection of hash centers to a predefined codebook $\mathcal{Z} \subset \{-1, 1\}^K$ with cardinality $|\mathcal{Z}| \ll 2^K$, significantly reducing the solution space. (2) **Dynamic Update**: We iteratively refine the hash centers in tandem with hash function training, thereby circumventing the combinatorial explosion associated with exhaustively exploring all possible configurations.

Experiment

Experimental Setups

Datasets We evaluate CRH on three widely used datasets covering both single- and multi-label retrieval tasks:

Stanford Cars (Krause et al. 2013): a single-label dataset with 196 vehicle categories, containing 8,144 training and 8,041 test images. Following (Wang et al. 2023), we use the training set for both model learning and the retrieval database, and the test set as queries.

Algorithm 1: Center-Reassigned Hashing (CRH)

```
Input: $K$, $C$, & ($M$, $\lambda$, $H$, $s$, $margin$).
Output: $f(\cdot)$ & $\{\mathbf{c}_c\}_{c=1}^C$.
Generate a hash codebook $\mathcal{Z}$ by randomly sampling $M$ binary vectors of dimension $K$;
Sample $C$ binary vectors from $\mathcal{Z}$ as initial class centers;
repeat
    Train the hash function $f(\cdot)$ by minimizing the objective in Eq. (4) for a specific number of epochs;
    for $h = 1$ to $H$ do
        Compute the cost matrix $\mathbf{L}^h$ using Eq. (7);
        Reassign sub-centers $\mathbf{c}_c^h$ via the greedy algorithm;
    end for
until convergence or max epochs reached;
Return the hash function $f(\cdot)$ and centers $\{\mathbf{c}_c\}_{c=1}^C$.
```

NABirds (Horn et al. 2015): a fine-grained single-label dataset with 555 bird species, split into 23,929 training and 24,633 test images. We adopt the same training/retrieval and query split as for Stanford Cars.

MS COCO (Lin et al. 2014): a multi-label dataset with 80 object categories. Following (Cao et al. 2017), we sample 5,000 query images, use 10,000 for training, and assign the remaining images to the retrieval database.

Metrics and Baselines We evaluate retrieval using mean Average Precision (mAP), reporting mAP@all for Stanford Cars and NABirds, and mAP@5,000 for MS COCO. Our method is compared against seven state-of-the-art deep hashing baselines: four center-based methods (CSQ (Yuan et al. 2020), OrthoHash (Hoe et al. 2021), MDS (Wang et al. 2023), SHC (Chen et al. 2025)) and three general approaches (HashNet (Cao et al. 2017), DTSH (Wang, Shi, and Kitani 2016), GreedyHash (Su et al. 2018)). For fairness, all use the same pre-trained ResNet-34 (He et al. 2016) backbone.

Implementation Details We optimize the model using Adam (Kingma and Ba 2015) with $(\beta_1, \beta_2) = (0.5, 0.999)$ and a weight decay of 10^{-5}. The margin $margin$ is set to 0.2 (Hoe et al. 2021), and the scale factor to $s = \sqrt{2}\log(C-1)$ (Zhang et al. 2019). The codebook size is fixed at $M = 2C$ (as shown in Figure 4). We employ a cosine annealing schedule (Loshchilov and Hutter 2017) with an initial learning rate of 10^{-4}. The hyperparameter λ is set to 0.1 for Stanford Cars and NABirds, and 0 for MS COCO (with more details in the *Supplementary*). The head dimension d is set to 16, 16, and 8 for the three datasets, respectively (as shown in Figure 4). By default, hash center reassignment uses the greedy algorithm, applied every epoch for the first 20 epochs and every 5 epochs thereafter (as shown in Figure 3). Training runs for 300 epochs on the single-label datasets and 30 epochs on MS COCO, with a batch size of 128.

Methods	**Stanford Cars** (mAP@all)			**NABirds** (mAP@all)			**MS COCO** (mAP@5K)		
	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
DTSH (Wang, Shi, and Kitani 2016)	57.48	71.32	76.20	23.46	32.91	44.01	79.14	81.66	82.79
HashNet (Cao et al. 2017)	25.99	40.73	50.84	11.84	20.26	29.47	70.62	74.36	76.92
GreedyHash (Su et al. 2018)	80.95	86.27	87.05	64.10	71.14	72.86	75.05	81.13	84.52
CSQ (Yuan et al. 2020)	79.13	85.77	87.17	65.70	70.62	74.07	72.44	81.32	85.30
OrthoHash (Hoe et al. 2021)	83.97	86.38	87.35	69.86	71.95	73.54	78.98	85.01	87.07
MDS (Wang et al. 2023)	85.33	86.49	87.76	65.14	71.47	74.08	73.53	79.98	82.57
SHC (Chen et al. 2025)	83.21	86.93	88.43	64.58	71.89	74.15	73.43	79.33	83.85
CRH (Ours)	**87.31**	**89.20**	**90.27**	**74.01**	**76.69**	**77.71**	**82.71**	**85.79**	**87.44**

Table 1: Comparison of retrieval performance (mAP, %) between our method and deep hashing baselines on three datasets at multiple bit lengths. Note that the mAP values in **Bold** and underlined indicate the best and second-best results, respectively.

Methods	**Stanford Cars** (mAP@all)			**NABirds** (mAP@all)			**MS COCO** (mAP@5K)		
	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
CRH	**87.31**	**89.20**	**90.27**	**74.01**	**76.69**	**77.71**	**82.71**	**85.79**	**87.44**
CRH-M	**87.31**	88.31	88.73	**74.01**	74.98	75.94	81.60	85.45	86.75
CRH-U	85.12	87.22	87.10	70.69	73.06	74.51	78.45	84.20	86.95

Table 2: Comparison of retrieval performance (mAP, %) between CRH and its ablated variants (CRH-M, CRH-U).

Results

The mAP results are summed in Table 1. Our CRH method consistently achieves state-of-the-art performance across all datasets, surpassing the strongest baselines by relative margins of 2.1%–2.6% (Stanford Cars), 4.8%–6.6% (NABirds), and 0.4%–4.5% (MS COCO). The substantial improvement on NABirds—a dataset characterized by intricate inter-class relationships due to its large category count—demonstrates the efficacy of our hash center update mechanism in modeling fine-grained semantic structures. On Stanford Cars, CRH delivers robust performance gains across all hash code lengths. For MS COCO, the most significant improvements occur at 16 bits, with diminishing returns observed for longer codes. Notably, DTSH and HashNet exhibit markedly inferior performance on single-label datasets with hundreds of categories, suggesting their tuple-similarity-based optimization objectives are less effective at capturing inter-class relationships in large-scale scenarios.

Ablation Studies

We conduct ablation studies to assess the contributions of two main components: center reassignment and the multi-head architecture. Specifically, we evaluate:

- **CRH-U**: Disables center reassignment and consequently the multi-head mechanism, resulting in a fixed-center approach similar to CSQ/OrthoHash;
- **CRH-M**: Retains reassignment but removes the multi-head mechanism (i.e., uses a single head, $H = 1$).

As illustrated in Table 2, CRH-M consistently outperforms CRH-U across nearly all settings, achieving average relative mAP gains of 1.9%, 3.08%, and 1.76% on Stanford Cars, NABirds, and MS COCO, respectively. These results underscore the effectiveness of dynamic center updates in capturing data semantics. Building on this, CRH achieves further improvements over CRH-M, with additional gains of 0.91%, 1.54%, and 0.85% on the same datasets, demonstrating the benefit of the multi-head design in refining hash centers for enhanced representation quality.

To test generalizability, we incorporate our update mechanism into CSQ, MDS, and OrthoHash (denoted $(\cdot)_U$), keeping their original codebooks (size C) without multi-heads. Table 3 shows consistent improvements in most cases, underscoring the broad applicability of our strategy for enhancing center-based hashing.

Robustness to Randomness

Our algorithm incorporates stochasticity through (1) random initialization of hash centers and (2) the greedy center update procedure. To assess their effects, we conduct:

- **Initialization Robustness**: 3 runs with different random center initializations (“Init”).
- **Update Robustness**: 3 runs with different random seeds but identical initializations (“Seed”).

As shown in Table 4, the consistently low standard deviations confirm the algorithm’s stability against these sources of randomness.

We also compare our greedy update (“Init”) with the Hungarian algorithm (“Init-H”). The greedy algorithm provides clear advantages: (1) slightly higher mAP (relative gains of 0.41%, 0.68%, and 0.26% across datasets), attributed to the stochasticity from its random class order which enables broader exploration of the solution space; and (2) lower computational complexity ($\mathcal{O}(HCM)$ vs. $\mathcal{O}(HC^2M)$).

Semantic Quality of Learned Hash Centers

To evaluate the semantic expressiveness of our learned hash centers, we design a quantitative framework using CLIP-ViT-L/14’s visual encoder (Radford et al. 2021) as a reference. For each class c, we compute its prototype vector $\mathbf{p}_c$ as a weighted average of sample features: $\mathbf{p}_c = \frac{1}{\sum_{\mathbf{x}\in\mathcal{X}_c} \frac{1}{\|\mathbf{y}_\mathbf{x}\|_1}} \sum_{\mathbf{x}\in\mathcal{X}_c} \frac{1}{\|\mathbf{y}_\mathbf{x}\|_1} \text{CLIP}(\mathbf{x})$. We then construct a reference similarity matrix $\mathbf{S}^r = (s^r_{ij})_{C\times C}$ with $s^r_{ij} =$

Methods	**Stanford Cars** (mAP@all)			**NABirds** (mAP@all)			**MS COCO** (mAP@5K)		
	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
CSQ	79.13	85.77	87.17	65.70	70.62	74.07	72.44	81.32	85.30
CSQ$_U$	**85.25**	**87.72**	**88.23**	**70.14**	**73.42**	**75.38**	**77.65**	**82.43**	**85.56**
MDS	85.33	86.49	87.76	65.14	71.47	74.08	73.53	79.98	82.57
MDS$_U$	**87.23**	**87.76**	**88.06**	**70.84**	**73.77**	**75.27**	**79.31**	**82.40**	**83.13**
OrthoHash	84.59	86.66	**87.35**	69.80	72.13	73.54	78.98	**85.01**	**87.07**
OrthoHash$_U$	**86.69**	**88.11**	**87.35**	**73.70**	**74.30**	**74.65**	**80.56**	84.24	86.33

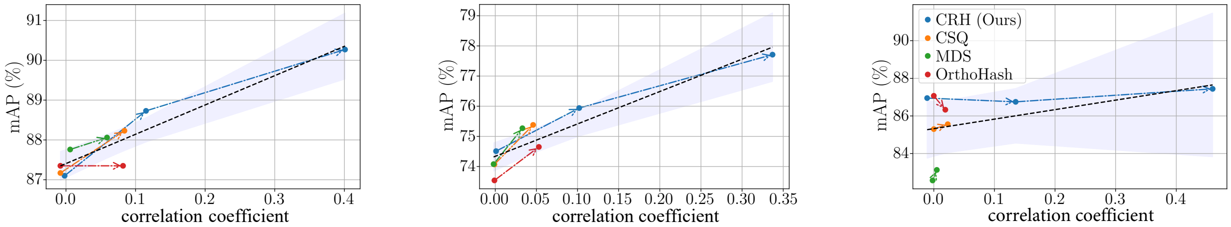
Table 3: Performance comparison (mAP, %) of hash-center-based methods with/without the proposed update mechanism.

Methods	**Stanford Cars** (mAP@all)			**NABirds** (mAP@all)			**MS COCO** (mAP@5K)		
	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
Seed	87.35 ± 0.03	89.01 ± 0.19	89.99 ± 0.24	73.98 ± 0.06	76.60 ± 0.08	77.72 ± 0.06	82.63 ± 0.08	85.72 ± 0.13	87.29 ± 0.12
Init	87.46 ± 0.11	88.89 ± 0.27	90.23 ± 0.06	74.09 ± 0.20	76.62 ± 0.17	77.83 ± 0.21	82.75 ± 0.03	85.74 ± 0.09	87.19 ± 0.20
Init-H	87.23 ± 0.10	88.80 ± 0.18	89.45 ± 0.06	73.45 ± 0.19	76.03 ± 0.02	77.53 ± 0.02	82.23 ± 0.31	85.73 ± 0.07	87.08 ± 0.10

Table 4: Performance impact of initialization randomness and update algorithms (mean±std mAP over 3 runs). "Seed": random update mechanism; "Init": varied center initializations; "Init-H": "Init" with Hungarian-algorithm updates.

Methods	**Stanford Cars**			**NABirds**			**MS COCO**		
	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
Init	0.004	-0.007	-0.002	-0.001	0.001	0.001	0.006	-0.025	-0.011
Learned	0.242	0.286	0.401	0.199	0.278	0.337	0.307	0.379	0.460
Learned-M	0.242	0.185	0.115	0.199	0.138	0.102	0.157	0.101	0.135
Random	0.000 ± 0.007	0.000 ± 0.008	-0.000 ± 0.007	-0.000 ± 0.003	-0.000 ± 0.003	-0.000 ± 0.003	0.001 ± 0.015	0.000 ± 0.016	0.000 ± 0.018
CSQ/OrthoHash	0.005	-0.005	-0.008	0.004	0.000	-0.001	0.002	0.008	0.000
MDS	0.002	0.004	0.006	0.001	-0.002	-0.002	-0.002	0.010	-0.002
SHC	0.022	0.046	0.011	0.001	0.011	-0.004	-0.002	0.010	-0.002

Table 5: Pearson correlation analysis between $\mathbf{S}^h$ and $\mathbf{S}^r$ under different center configurations: initialized ("Init"), learned (multi-head "Learned"/single-head "Learned-M"), and random assignments ("Random", mean±std over 1,000 runs). Besides, the results of the four baseline methods (CSQ, OrthoHash, MDS, and SHC) are also included.



(a) Stanford Cars

(b) NABirds

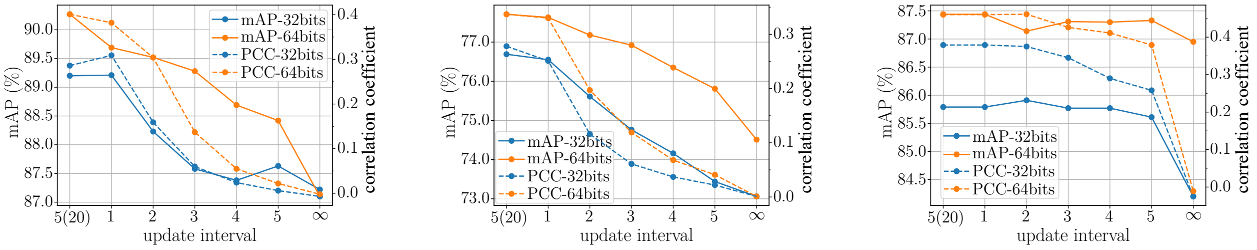
(c) MS COCO

Figure 2: mAP vs. PCC (64-bit) on different datasets. Arrows link baseline variants ("original" → "updated", e.g., MDS → MDS$_U$) and **CRH-U** → **CRH-M** → **CRH**. Regression lines with 95% confidence intervals indicate the linear trend.

$\cos(\mathbf{p}_i, \mathbf{p}_j)$, and a corresponding hash center similarity matrix $\mathbf{S}^h = (s^h_{ij})_{C\times C}$ based on $s^h_{ij} = \cos(\mathbf{c}_i, \mathbf{c}_j)$. The alignment between learned hash centers and reference semantics is quantified by the **Pearson correlation coefficient (PCC)** computed over the strictly upper triangular elements of $\mathbf{S}^r$ and $\mathbf{S}^h$. Higher PCC indicates stronger semantic alignment.

As shown in Table 5, untrained centers yield near-zero correlation (row 1). In contrast, our multi-head design achieves average PCCs of 0.310, 0.271, and 0.382 across the three datasets (row 2), validating its ability to learn semantically meaningful hash centers. The single-head variant (row 3) shows notably lower correlations (0.181, 0.146, and 0.131), highlighting the effectiveness of the multi-head mechanism. Random centers (row 4) exhibit no semantic structure. Among baselines, CSQ, OrthoHash, and MDS—with randomly assigned centers —show no semantic alignment, while SHC incorporates semantics but still yields low correlations, indicating limited ability to capture fine-grained class relationships.

Finally, we analyze the link between semantic alignment and retrieval. Figure 2 plots mAP against PCC for three baselines and **CRH** (64-bit). A clear linear trend emerges—especially on single-label datasets—showing that more expressive hash centers lead to better retrieval. This trend is weaker on MS COCO. Notably, CSQ and OrthoHash generate hash centers with nearly identical inter-center distances on this dataset, impeding semantic capture through reassignment and hence limiting performance gains. CRH

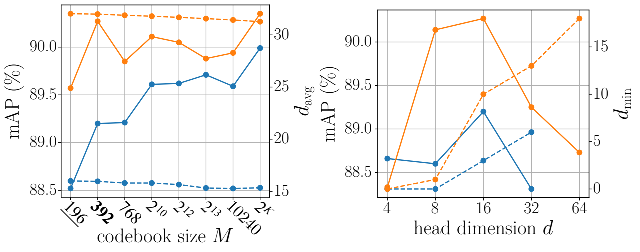


(a) Stanford Cars

(b) NABirds

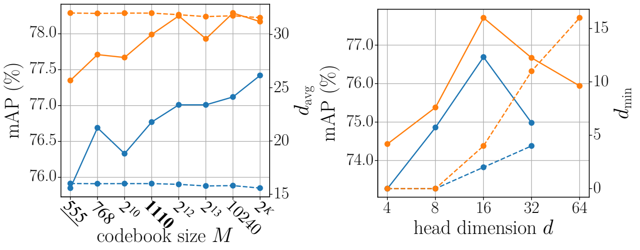
(c) MS COCO

Figure 3: Impact of hash center update frequency on mAP/PCC across datasets. “5 (20)” indicates updates every epoch for the first 20 epochs, then every 5; “interval=2” denotes updates every 2 epochs (similarly for other values); “∞” means no updates.



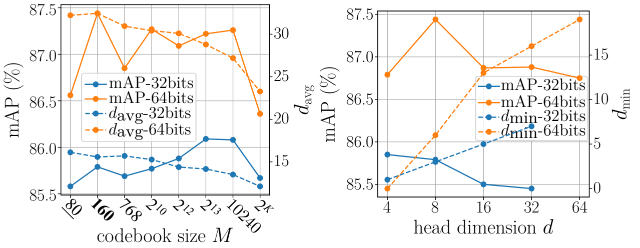
(a) Stanford Cars (M)

(b) Stanford Cars (d)



(c) NABirds (M)

(d) NABirds (d)



(e) MS COCO (M)

(f) MS COCO (d)

Figure 4: mAP w.r.t. codebook size M (left) and head dimension d (right) on three datasets. Underscore and **bold** ticks denote $M = C$ and $M = 2C$.

overcomes this through larger randomized codebooks and multi-head design, enhancing diversity and semantic fidelity.

Impact of Codebook Size and Head Dimension

We evaluate how codebook size M and head dimension d affect retrieval performance. Figure 4 shows mAP alongside the minimum ($d_{\min}$) and average (d_{avg}) distances between learned hash centers under varying configurations at 32/64-bit lengths. Note that for $M = 2^K$ (i.e., $\mathcal{Z} = \{-1, 1\}^K$), explicitly constructing $\mathcal{Z}$ is infeasible for large K, so we approximate reassignment: for each class c, we set $\mathbf{z} = \mathrm{sign}(\sum_{\mathbf{x}\in\mathcal{X}_c} \mathrm{sign}(\mathbf{h}_{\mathbf{x}})/\|\mathbf{y}_{\mathbf{x}}\|_1)$ as its center if unassigned; otherwise the closest unassigned $\mathbf{z}' \in \mathcal{Z}$ to $\mathbf{z}$ is assigned. For M, larger codebooks (vs. $M = C$) generally improve mAP but increase computation. On single-label datasets, d_{avg} remains near $K/2$ and declines slowly with M, while on MS COCO it drops significantly, achieving optimal performance at moderate M. We therefore set $M = 2C$ to balance performance and efficiency. For d, best performance occurs when using the smallest power-of-2 dimension satisfying $d \geq \log_2 M$ (yielding 16, 16, 8 for $M = 2C$ across datasets), maximizing the number of heads while preventing codebook collisions to enhance semantic expressiveness. Conversely, excessively small d causes collisions that reduce $d_{\min}$ and degrade performance.

Impact of Centers’ Update Frequency

Figure 3 illustrates how the update interval (i.e., epochs between center updates) affects mAP and PCC (32/64-bit). Longer intervals reduce PCC, indicating weaker semantic alignment, and lead to lower mAP. Although MS COCO is less sensitive than the single-label datasets, omitting updates (∞) still results in substantial performance drops. These results confirm that more frequent updates generally improve performance. When training on single-label datasets with “interval=1”, we observed that roughly 20% of class centers were updated per step in the first 20 epochs on average, dropping below 1% afterward, suggesting convergence. Thus, we increased the interval to 5 after 20 epochs, maintaining performance while significantly reducing computation.

Conclusion

We propose Center-Reassigned Hashing (CRH), an end-to-end deep hashing framework that jointly optimizes hash centers and hash function. By integrating a hash codebook with a multi-head mechanism, CRH dynamically reassigns hash centers to categories, allowing them to adapt to data distributions for enhanced semantic representation—eliminating the need for auxiliary networks or iterative center optimization. Extensive experiments demonstrate that CRH learns semantically discriminative hash centers and achieves state-of-the-art retrieval performance.

Acknowledgments

This work is supported in part by National Natural Science Foundation of China (NSFC, Grant No. 62372054, 62006005) and National Key Research and Development Program of China (Grant No. 2022YFC3302200).

References

Cao, Y.; Long, M.; Liu, B.; and Wang, J. 2018. Deep Cauchy Hashing for Hamming Space Retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1229–1237.

Cao, Z.; Long, M.; Wang, J.; and Yu, P. S. 2017. HashNet: Deep Learning to Hash by Continuation. In *IEEE International Conference on Computer Vision*, 5609–5618.

Charikar, M. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing*, 380–388.

Chen, L.; Liu, R.; Zhou, Y.; Ma, X.; Chen, Y.; and Zhang, D. 2025. Deep Hashing with Semantic Hash Centers for Image Retrieval. *ACM Transaction on Information Systems*, 1–40.

Fan, L.; Ng, K. W.; Ju, C.; Zhang, T.; and Chan, C. S. 2020. Deep Polarized Network for Supervised Learning of Accurate Binary Hashing Codes. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, 825–831.

Gionis, A.; Indyk, P.; and Motwani, R. 1999. Similarity Search in High Dimensions via Hashing. In *Proceedings of 25th International Conference on Very Large Data Bases*, 518–529.

Gong, Y.; Lazebnik, S.; Gordo, A.; and Perronnin, F. 2013. Iterative Quantization: A Procrustean Approach to Learning Binary Codes for Large-Scale Image Retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(12): 2916–2929.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 770–778.

Hoe, J. T.; Ng, K. W.; Zhang, T.; Chan, C. S.; Song, Y.; and Xiang, T. 2021. One Loss for All: Deep Hashing with a Single Cosine Similarity based Learning Objective. In *Advances in Neural Information Processing Systems 34*, 24286–24298.

Horn, G. V.; Branson, S.; Farrell, R.; Haber, S.; Barry, J.; Ipeirotis, P.; Perona, P.; and Belongie, S. J. 2015. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 595–604.

Indyk, P.; and Motwani, R. 1998. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing*, 604–613.

Jiang, Q.; and Li, W. 2017. Deep Cross-Modal Hashing. In *IEEE Conference on Computer Vision and Pattern Recognition*, 3270–3278.

Jiang, Q.; and Li, W. 2018. Asymmetric Deep Supervised Hashing. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 3342–3349.

Jose, A.; Filbert, D.; Rohlfing, C.; and Ohm, J. 2022. Deep Hashing with Hash Center Update for Efficient Image Retrieval. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 4773–4777.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In Bengio, Y.; and LeCun, Y., eds., *3rd International Conference on Learning Representations*.

Kong, W.; and Li, W. 2012. Isotropic Hashing. In *Advances in Neural Information Processing Systems 25*, 1655–1663.

Krause, J.; Stark, M.; Deng, J.; and Fei-Fei, L. 2013. 3D Object Representations for Fine-Grained Categorization. In *IEEE International Conference on Computer Vision Workshops*, 554–561.

Kuhn, H. W. 1955. The Hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2): 83–97.

Kulis, B.; and Darrell, T. 2009. Learning to Hash with Binary Reconstructive Embeddings. In *Advances in Neural Information Processing Systems 22*, 1042–1050.

Lai, H.; Pan, Y.; Liu, Y.; and Yan, S. 2015. Simultaneous feature learning and hash coding with deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 3270–3278.

Li, Q.; Sun, Z.; He, R.; and Tan, T. 2017. Deep Supervised Discrete Hashing. In *Advances in Neural Information Processing Systems 30*, 2482–2491.

Li, W.; Wang, S.; and Kang, W. 2016. Feature Learning Based Deep Supervised Hashing with Pairwise Labels. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 1711–1717.

Lin, T.; Maire, M.; Belongie, S. J.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft COCO: Common Objects in Context. In *Computer Vision - ECCV 2014 - 13th European Conference*, 740–755.

Liu, W.; Wang, J.; Ji, R.; Jiang, Y.; and Chang, S. 2012. Supervised hashing with kernels. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2074–2081.

Loshchilov, I.; and Hutter, F. 2017. SGDR: Stochastic Gradient Descent with Warm Restarts. In *5th International Conference on Learning Representations*.

Lu, J.; Wang, H.; Zhou, J.; Yan, M.; and Wen, J. 2022. Deep asymmetric hashing with dual semantic regression and class structure quantization. *Inf. Sci.*, 589: 235–249.

Norouzi, M.; and Fleet, D. J. 2011. Minimal Loss Hashing for Compact Binary Codes. In *Proceedings of the 28th International Conference on Machine Learning*, 353–360.

Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; Krueger, G.; and Sutskever, I. 2021. Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of the 38th International Conference on Machine Learning*, 8748–8763.

Su, S.; Zhang, C.; Han, K.; and Tian, Y. 2018. Greedy Hash: Towards Fast Optimization for Accurate Hash Coding in CNN. In *Advances in Neural Information Processing Systems 31*, 806–815.

Wang, L.; Pan, Y.; Lai, H.; and Yin, J. 2022. Image Retrieval with Well-Separated Semantic Hash Centers. In *16th Asian Conference on Computer Vision*, 716–731.

Wang, L.; Pan, Y.; Liu, C.; Lai, H.; Yin, J.; and Liu, Y. 2023. Deep Hashing with Minimal-Distance-Separated Hash Centers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 23455–23464.

Wang, X.; Shi, Y.; and Kitani, K. M. 2016. Deep Supervised Hashing with Triplet Labels. In *13th Asian Conference on Computer Vision*, 70–84.

Weiss, Y.; Torralba, A.; and Fergus, R. 2008. Spectral Hashing. In *Advances in Neural Information Processing Systems 21*, 1753–1760.

Xiao, X.; Cao, S.; Wang, L.; Cheng, S.; and Yuan, E. 2024. Deep hashing image retrieval based on hybrid neural network and optimized metric learning. *Knowl. Based Syst.*, 284: 111336.

Yang, H.; Lin, K.; and Chen, C. 2018. Supervised Learning of Semantics-Preserving Hash via Deep Convolutional Neural Networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(2): 437–451.

Yuan, L.; Wang, T.; Zhang, X.; Tay, F. E. H.; Jie, Z.; Liu, W.; and Feng, J. 2020. Central Similarity Quantization for Efficient Image and Video Retrieval. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3080–3089.

Zhang, X.; Zhao, R.; Qiao, Y.; Wang, X.; and Li, H. 2019. AdaCos: Adaptively Scaling Cosine Logits for Effectively Learning Deep Face Representations. In *IEEE Conference on Computer Vision and Pattern Recognition*, 10823–10832.