Pairwise-Label-Based Deep Incremental Hashing with Simultaneous Code Expansion

Dayan Wu[1], Qinghang Su[1,2], Bo Li[1*], Weiping Wang[1]

[1]Institute of Information Engineering, Chinese Academy of Sciences
[2]School of Cyber Security, University of Chinese Academy of Sciences
{wudayan, suqinghang, libo, wangweiping}@iie.ac.cn

Abstract

Deep incremental hashing has become a subject of considerable interest due to its capability to learn hash codes in an incremental manner, eliminating the need to generate codes for classes that have already been learned. However, accommodating more classes requires longer hash codes, and regenerating database codes becomes inevitable when code expansion is required. In this paper, we present a unified deep hash framework that can simultaneously learn new classes and increase hash code capacity. Specifically, we design a triple-channel asymmetric framework to optimize a new CNN model with a target code length and a code projection matrix. This enables us to directly generate hash codes for new images, and efficiently generate expanded hash codes for original database images from the old ones with the learned projection matrix. Meanwhile, we propose a pairwise-label-based incremental similarity-preserving loss to optimize the new CNN model, which can incrementally preserve new similarities while maintaining the old ones. Additionally, we design a double-end quantization loss to reduce the quantization error from new and original query images. As a result, our method efficiently embeds both new and original similarities into the expanded hash codes, while keeping the original database codes unchanged. We conduct extensive experiments on three widely-used image retrieval benchmarks, demonstrating that our method can significantly reduce the time required to expand existing database codes, while maintaining state-of-the-art retrieval performance.

Introduction

Hashing can encode visual data into compact binary codes so that visually similar samples are mapped into similar binary codes. Traditional hashing methods using hand-crafted features, for instance, Locality Sensitive Hashing (LSH) (Gionis, Indyk, and Motwani 1999), Spectral Hashing (SH) (Weiss, Torralba, and Fergus 2009) and Iterative Quantization (Gong and Lazebnik 2011), perform worse in preserving the semantic similarities between visual samples compared to deep hashing methods, which incorporate deep neural networks into the generation of hash codes for large-scale visual search. The deep neural networks have powerful representation abilities to maintain the underlying semantics of visual data. As a result, deep hashing has shown much better retrieval performance than traditional hashing methods. Recently, a variety of deep hashing methods have been presented, which can be classified into two categories, i.e., symmetric hashing methods (Luo et al. 2019; Zhuang et al. 2016; Tang et al. 2018; Zhang et al. 2018; Cui et al. 2019; Jin et al. 2020; Deng et al. 2018; Zhang et al. 2019; Doan, Yang, and Li 2022; Cui et al. 2022; Tu et al. 2022; Zhang et al. 2021b, 2017; Zhao et al. 2021; Zhang et al. 2021c, 2020; Liu et al. 2018b) and asymmetric hashing methods (Chen et al. 2019; Wu et al. 2019; Jiang and Li 2018; Wu et al. 2023; Song et al. 2022; Luo et al. 2018; Gu et al. 2019; Zhang et al. 2021a).

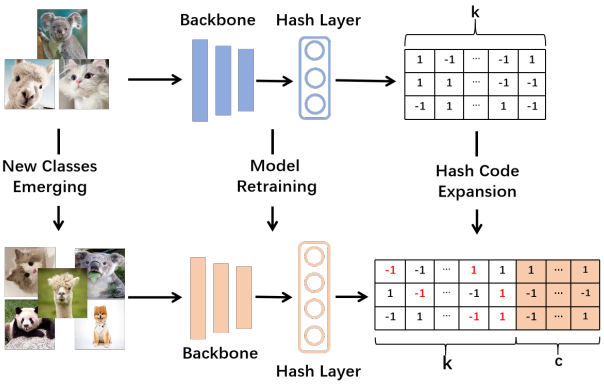

Figure 1: The framework of the conventional deep hashing methods for new classes learning and code expansion. When new classes emerge, conventional methods have to retrain the models with new hash layers and new training images. The new database images are required to be fed into the new models to generate hash codes with target code length again.

The explosive growth of web and sensor images has led to a large number of new semantic concepts, imposing significant challenges to conventional deep hashing methods. To accommodate new classes, conventional deep hashing methods require retraining the models and re-indexing the entire database codes, which is time-consuming. To tackle this problem, deep incremental hashing methods have been proposed (Wu et al. 2019; Mandal and Biswas 2020; Tian, Ng, and Xu 2023). These methods can incrementally learn hash codes for training images from new categories while hold-

*Corresponding author.

ing the original ones invariant. However, with the constant emergence of new classes, the feature space of these methods becomes crowded, which can adversely affect retrieval performance. To increase capacity, as illustrated in Figure 1, similar to conventional deep hashing methods, deep incremental hashing methods can only retrain the CNN model with a target code length and re-index the entire database codes. As the number of new concepts increases, the limited capacity of hash codes may force deep incremental hashing methods to degrade into conventional deep hashing methods.

In this paper, we propose a unified deep hashing framework, called Code Expansion enabled Deep Incremental Hashing (CEDIH), to simultaneously achieve fast hash code expansion and new class learning. CEDIH allows for fast code expansion and efficient class learning without changing the original database codes. The main challenge in CEDIH is how to incrementally generate hash codes for new classes with target code length and new bits for original classes, while preserving three types of similarities: the similarities between new images *(new-new similarities)*, the similarities between original images *(original-original similarities)*, and the similarities between original and new images *(original-new similarities)*. Note that during the process of preserving original-original similarities and original-new similarities, the original database codes remain unchanged. An overview of the proposed framework is illustrated in Figure 2. Specifically, CEDIH adopts a triple-channel asymmetric design to optimize a new CNN model with a target code length, a code projection matrix, and binary hash codes for new training images in an end-to-end manner. The new CNN model generates expanded hash codes for query images and new database images. By using the learned code projection matrix, original database codes can be quickly expanded through simple matrix multiplication. To optimize the triple-channel framework, we design a pairwise-label-based incremental similarity-preserving loss function. This function can maintain the three types of similarities. In addition, the CEDIH quantization errors come from both new and original query images. Therefore, we design a double-end quantization loss to reduce the quantization errors. The main contribution of this work can be summarized as follows.

- We propose a novel deep hashing framework, named *Code Expansion enabled Deep Incremental Hashing* (CEDIH), which targets at acquiring the capability of learning new classes while enlarging the capacity of hash codes simultaneously. To the best of our knowledge, CEDIH is the first deep hashing approach that can do both simultaneously.
- We design a pairwise-label-based incremental similarity-preserving loss to guide the learning of a new CNN model and the code projection matrix in an end-to-end manner. This allows for the efficient embedding of both the new and original similarities into the expanded database codes, while keeping the original similarities unchanged. We also propose a double-end quantization loss function that separately considers the quantization errors introduced by new and original query images. This further improves the retrieval performance.
- Extensive experiments demonstrate that the proposed approach can significantly decrease code expansion time when accommodating new concepts, with almost no loss in retrieval accuracy compared to state-of-the-art methods. We validate that our CEDIH is compatible with a variety of backbone deep hashing methods.

Related Work

Conventional deep hashing methods can be classified into three categories based on the form of supervision: pointwise-label-based methods (Fan et al. 2020; Yang, Lin, and Chen 2018; Shen et al. 2019; Hoe et al. 2021; Yuan et al. 2020; Liu et al. 2019), pairwise-label-based methods (Liu et al. 2016; Li, Wang, and Kang 2016; Wu et al. 2017; Cao et al. 2017; Xia et al. 2014; Lai et al. 2015; Cakir, He, and Sclaroff 2018; Cakir et al. 2019; Shen et al. 2017; Jiang and Li 2018), and listwise-label-based methods (Zhao et al. 2015; Yao et al. 2016; Liu et al. 2018a). Pointwise-label-based methods optimize hash codes with classification objectives or pre-defined targets. CSQ (Yuan et al. 2020) and OrthoHash (Hoe et al. 2021) are representative pointwise-label-based methods that utilize the Hadamard matrix or Bernoulli sampling to generate hash centers. Pairwise-label-based methods try to push similar pairs together while dissimilar ones apart only with a similarity matrix. DSH (Liu et al. 2016) and DPSH (Li, Wang, and Kang 2016) are two representative deep hashing methods that design effective loss functions with pairwise labels. DAPH (Shen et al. 2017) integrates feature learning and asymmetric hash function learning into the end-to-end deep learning framework. Different from DAPH, ADSH (Jiang and Li 2018) trains a CNN model only for query images and directly generates the final binary database codes during the optimization process. Listwise-label-based methods learn hash codes by preserving the supervised ranking list information, which is calculated based on the semantic labels. DSRH (Zhao et al. 2015) learns hash functions by preserving semantic similarity between multi-label images.

Deep incremental hashing methods aim to efficiently update deep hashing models when new classes emerge. DIHN (Wu et al. 2019) is the first deep incremental hashing method that can incrementally generate hash codes for images from new concepts while holding the original ones unchanged. iCMH (Mandal and Biswas 2020) proposes a novel incremental cross-modal hashing algorithm, whose model parameters can be updated without suffering from catastrophic forgetting. DIH (Tian, Ng, and Xu 2023) considers concept drift problems in the incremental setting.

Code expansion and compression oriented deep hashing methods take into account the scalability of deep hashing models. Conventional deep hashing methods can only train models with pre-defined code length, and there are no theories to guide how to set the proper code length. If the code length is too long, both storage and computational efficiency would be affected, while retrieval accuracy would not be guaranteed. CCDH (Zhao et al. 2020) is the first deep

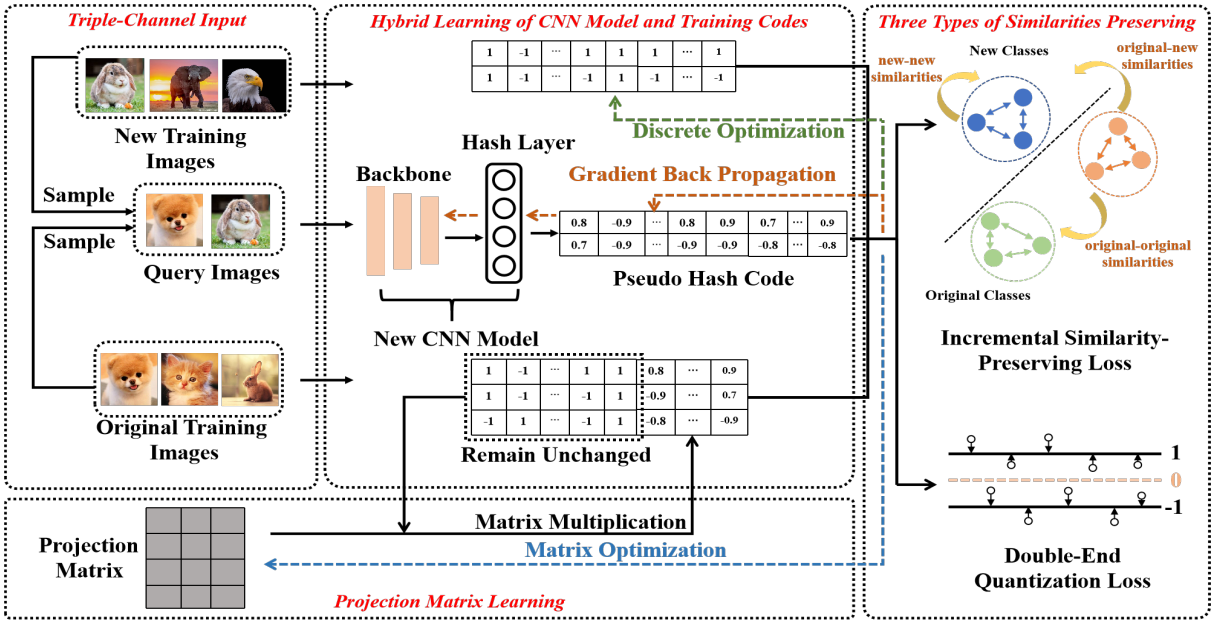

Figure 2: The triple-channel asymmetric framework of CEDIH. A new CNN model with target code length is trained as the hash function for query images and new database images. The hash codes of new training images are directly generated through discrete optimization. To generate new database codes from the original database images, we learn a projection matrix through simple matrix multiplication while keeping the original database unchanged. Best viewed in color.

hashing method designed to compress hash codes efficiently. CEDH (Wu et al. 2022) is the first deep hashing network attempting to expand hash codes efficiently. Neither CCDH nor CEDH requires regenerating the entire database's hash codes through CNN models, which significantly improves the scalability of deep hashing models.

However, most of the aforementioned deep hashing methods focus on learning discriminative binary codes for a fixed dataset with a specific code length. DIHN (Wu et al. 2019) ignores the need for capacity expansion of hash codes when new concepts are learned. CEDH (Wu et al. 2022) cannot learn new classes when expanding hash codes. Growbit (Mandal, Annadani, and Biswas 2019) attempts to integrate new concepts and learn additional bits in a two-stage process for cross-modal retrieval. Nevertheless, this approach cannot scale to single-modal image retrieval due to its cross-modal setting and two-stage framework. To achieve satisfactory performance, new bits learning and hash function optimization must occur simultaneously. Though our method and CEDH (Wu et al. 2022) share the similar idea of expanding original database codes with a projection matrix, our primary challenge is how to embed the three types of similarities into the new bits for original classes and the new hash codes for new classes, while holding the original database codes unchanged.

Methodology

Problem Definition

Consider the case that we have $\mathbf{M}$ database images of original classes $\mathbf{L}_{ori}$, and database hash codes $\mathbf{B}_{ori} = \{\overline{b}_i\}_{i=1}^{\mathbf{M}} \in \{-1,+1\}^{\mathbf{M}\times k}$ are trained, where k is the code length. Assume that a set of $\mathbf{N}$ incremental images emerges, in which each image belongs to a new class set $\mathbf{L}_{inc}$. Note that $\mathbf{L}_{ori}$ and $\mathbf{L}_{inc}$ are assumed to be disjoint, i.e. $\mathbf{L}_{ori} \cap \mathbf{L}_{inc} = \varnothing$. To accommodate more categories, we need to use longer hash codes $\mathbf{B} = \{b_i\}_{i=1}^{\mathbf{M}+\mathbf{N}} \in \{-1,+1\}^{(\mathbf{M}+\mathbf{N})\times k'}$ for all images, where $k' = k + c$ and c is the length of new bits. We aim to directly learn the new hash bits $\mathbf{B}_{new} = \{\tilde{b}_i\}_{i=1}^{\mathbf{M}} \in \{-1,+1\}^{\mathbf{M}\times c}$ for original database images, while the hash codes $\mathbf{B}_{inc} = \{b_i\}_{i=\mathbf{M}+1}^{\mathbf{M}+\mathbf{N}} \in \{-1,+1\}^{N\times k'}$ of new emerging images are learned from scratch.

Besides, assuming we have m original training images and n new training images, their indices are denoted as $\Psi = \{\mathbf{O}_1, ..., \mathbf{O}_m\}$ and $\Phi = \{\mathbf{I}_1, ..., \mathbf{I}_n\}$ respectively. To construct the query set $\mathbf{Q} = \{a_i\}_{i=1}^{q}$, we randomly sample q images from the training set. A CNN model is adopted as the hash function to generate their hash codes $\mathbf{B_Q} = \{\hat{b}_i\}_{i=1}^{q} \in \{-1,+1\}^{q\times k'}$. We use $\psi = \{o_1, ..., o_{q^*}\} \subset \Psi$ and $\phi = \{i_1, ..., i_{q'}\} \subset \Phi$ to denote the indices of the sampled query images of original and incremental classes. The corresponding indices of ψ and ϕ in $\mathbf{Q}$ are denoted as ψ^* and ϕ^*. The similarity matrix $\mathbf{S} \in \{-1,+1\}^{q\times(m+n)}$ between query and training images is available during training. $\mathbf{S}_{ij} = 1$ indicates that query image a_i and training image d_j are similar, while $\mathbf{S}_{ij} = -1$ indicates the opposite. Furthermore, $\mathbf{S}^{\Psi} \in \{-1,+1\}^{q\times m}$ is the similarity matrix between query and original training images, while $\mathbf{S}^{\Phi} \in \{-1,+1\}^{q\times n}$ is the similarity matrix between the query and new training images.

Proposed Framework

Our goal is to embed both new and original similarities into the expanded hash codes while keeping the original database codes unchanged. To this end, we decouple the learning of the CNN model and the training codes of new and original images. Specifically, we design a triple-channel asymmetric framework to train the CNN model with sampled new and original training images for query images, directly optimize discrete codes for new training images, and learn a projection matrix to expand existing database codes. As shown in Figure 2, CEDIH consists of four parts: triple-channel input part, projection matrix learning part, hybrid learning part of CNN model and training codes, and three types of similarities preserving part.

Incremental Similarity-Preserving Loss

Incremental similarity-preserving loss aims to incrementally preserve new-new and original-new similarities while maintaining original-original ones, which can be formulated as:

$$\mathbf{R} = \|\mathbf{B_Q}(\mathbf{B}^{\Psi})^{\mathbf{T}} - k'(\mathbf{S}^{\Psi})\|_2^2 + \|\mathbf{B_Q}(\mathbf{B}^{\Phi})^{\mathbf{T}} - k'(\mathbf{S}^{\Phi})\|_2^2 \quad (1)$$

where $\mathbf{B_Q} = \{sign(f_i(\theta))\}_{i=1}^q$, and $f(\cdot)$ is the output of the last fully connected layer and θ denotes the parameters of the new CNN model. The first term aims at preserving similarities between query images and original training images, while the second term aims at preserving similarities between query images and new training images. Note that query images are sampled from new training images and original training images. Therefore, the similarity preserving loss can preserve three types of similarities. Since $\mathbf{B}^{\Psi}$ is generated by concatenating the fixed $\mathbf{B}^{\Psi}_{ori}$ and the learned $\mathbf{B}^{\Psi}_{new}$, the loss function can be reformulated as :

$$\begin{aligned}\mathbf{R} = &\|\mathbf{B}_{\mathbf{Q}_1}(\mathbf{B}^{\Psi}_{ori})^{\mathbf{T}} + \mathbf{B}_{\mathbf{Q}_2}(\mathbf{B}^{\Psi}_{new})^{\mathbf{T}} - k'\mathbf{S}^{\Psi}\|_2^2 \\ &+ \|\mathbf{B_Q}(\mathbf{B}^{\Phi})^{\mathbf{T}} - k'\mathbf{S}^{\Phi}\|_2^2\end{aligned} \quad (2)$$

where $\mathbf{B}_{\mathbf{Q}_1} = \{sign(f_i(\theta_1))\}_{i=1}^q$ denotes the first k columns of $\mathbf{B_Q}$, while $\mathbf{B}_{\mathbf{Q}_2} = \{sign(f_i(\theta_2))\}_{i=1}^q$ represents the last c columns. θ_1 and θ_2 share the parameters of the backbone and only differ in the last fully connected layer. As the $sign(\cdot)$ function is difficult to directly optimize. Hence, we approximate it with the function $tanh(\cdot)$, bringing the new formulation by replacing $\mathbf{B_Q}$ with $\mathbf{U}$:

$$\begin{aligned}\mathbf{R} = &\|\mathbf{U}_1(\mathbf{B}^{\Psi}_{ori})^{\mathbf{T}} + \mathbf{U}_2(\mathbf{B}^{\Psi}_{new})^{\mathbf{T}} - k'\mathbf{S}^{\Psi}\|_2^2 \\ &+ \|\mathbf{U}(\mathbf{B}^{\Phi})^{\mathbf{T}} - k'\mathbf{S}^{\Phi}\|_2^2\end{aligned} \quad (3)$$

where $\mathbf{U} = \{u_i = tanh(f_i(\theta))\}_{i=1}^q$, $\mathbf{U}_1 = \{u_{i1} = tanh(f_i(\theta_1))\}_{i=1}^q$ and $\mathbf{U}_2 = \{u_{i2} = tanh(f_i(\theta_2))\}_{i=1}^q$. Note that $\mathbf{B}^{\Psi}_{new}$ is generated from $\mathbf{B}^{\Psi}_{ori}\mathbf{W}, \mathbf{W} \in \mathbf{R}^{k\times c}$.

Double-End Quantization Loss

Double-end quantization loss aims at reducing quantization errors produced by both new and original query images. As shown in Figure 3, the quantization errors come from four parts: original bits of original query images (Term 1), new bits of original query images (Term 2), and hash codes of

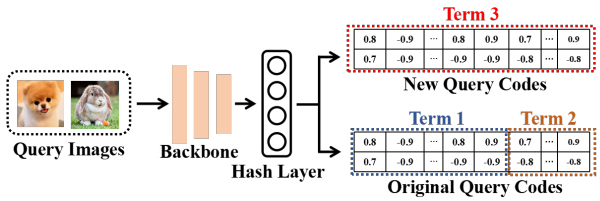

Figure 3: The illustration of the double-end quantization loss.

new query images (Term 3). The quantization loss can be formulated as:

$$\begin{aligned}\mathbf{Q} = &\underbrace{\|\mathbf{U}_1^{\psi^*} - \mathbf{B}^{\psi}_{ori}\|_2^2}_{\text{Term 1}} + \underbrace{\|\tilde{\mathbf{U}}_2^{\Psi} - \mathbf{Z}\|_2^2 + \|\mathbf{Z} - \mathbf{B}^{\Psi}_{ori}\mathbf{W}\|_2^2}_{\text{Term 2}} \\ &+ \underbrace{\|\mathbf{U}^{\phi^*} - \mathbf{B}^{\phi}\|_2^2}_{\text{Term 3}}\end{aligned} \quad (4)$$

where $\mathbf{Z} = \{z_i\}_{i=1}^m \in \{-1,+1\}^{m\times c}$, $\tilde{\mathbf{U}}_2^{\Psi} = \{\tilde{u}_i | i \in \Psi\} \in [-1,+1]^{m\times c}$, and $\tilde{u}_i$ is defined as:

$$\tilde{u}_i = \begin{cases} u_{i^*2}, & \text{if } i^* \in \psi^* \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

Note that a one-to-one mapping exists between i and i^* if $i^* \in \psi^*$. The final loss function is formed as:

$$\begin{aligned}&\min_{\theta,\mathbf{W},\mathbf{B}^{\Phi},\mathbf{Z}} \mathbf{L} = \mathbf{R} + \lambda\mathbf{Q} \\ &s.t.\ \mathbf{B}^{\Phi} \in \{-1,+1\}^{n\times k'}, \mathbf{Z} \in \{-1,+1\}^{m\times c}.\end{aligned} \quad (6)$$

where $\lambda > 0$ is a hyper-parameter.

Optimization

We adopt an alternating strategy to optimize the parameters θ, $\mathbf{W}$, $\mathbf{B}^{\Phi}$, and $\mathbf{Z}$ in Eqn. (6). We optimize one parameter with the other three fixed.

θ-step: When $\mathbf{W}$, $\mathbf{B}^{\Phi}$, and $\mathbf{Z}$ are fixed, we use the standard back-propagation algorithm to optimize θ.

W-step: We learn $\mathbf{W}$ with other parameters fixed. The problem becomes:

$$\begin{aligned}\min_{\mathbf{W}} \mathbf{L} = &\|\mathbf{U}_1(\mathbf{B}^{\Psi}_{ori})^{\mathbf{T}} + \mathbf{U}_2(\mathbf{B}^{\Psi}_{ori}\mathbf{W})^{\mathbf{T}} - k'\mathbf{S}^{\Psi}\|_2^2 \\ &+ \lambda\|\mathbf{Z} - \mathbf{B}^{\Psi}_{ori}\mathbf{W}\|_2^2\end{aligned} \quad (7)$$

Let $\frac{\partial \mathbf{L}}{\partial \mathbf{W}} = 0$, we can get the closed-form solution of $\mathbf{W}$:

$$\mathbf{W} = [(\mathbf{B}^{\Psi}_{ori})^{\mathbf{T}}\mathbf{B}^{\Psi}_{ori} + \gamma\mathbf{I}]^{-1}\mathbf{J}[\mathbf{U}_2^{\mathbf{T}}\mathbf{U}_2 + \lambda\mathbf{I}]^{-1} \quad (8)$$

where $\mathbf{J} = (\mathbf{B}^{\Psi}_{ori})^{\mathbf{T}}(\mathbf{P}^{\mathbf{T}}\mathbf{U}_2 + \lambda\mathbf{Z})$, $\mathbf{P} = k'\mathbf{S}^{\Psi} - \mathbf{U}_1(\mathbf{B}^{\Psi}_{ori})^{\mathbf{T}}$ and $\gamma > 0$ is a hyper-parameter.

Z-step: When fixing θ, $\mathbf{W}$ and $\mathbf{B}^{\Phi}$, we can rewrite Eqn. (6) as:

$$\begin{aligned}&\min_{\mathbf{Z}} \mathbf{L} = \|\tilde{\mathbf{U}}_2^{\Psi} - \mathbf{Z}\|_2^2 + \|\mathbf{Z} - \mathbf{B}^{\Psi}_{ori}\mathbf{W}\|_2^2 \\ &s.t.\ \mathbf{Z} \in \{-1,+1\}^{m\times c}\end{aligned} \quad (9)$$

Finally, we have the solution:

$$\mathbf{Z} = sign(\tilde{\mathbf{U}}_2^{\Psi} + \mathbf{B}_{ori}^{\Psi}\mathbf{W}) \tag{10}$$

$\mathbf{B}^{\Phi}$-step: Given the fixed θ, $\mathbf{W}$ and $\mathbf{Z}$, the objective in problem (6) can be rewritten into the formulation of:

$$\begin{aligned} \min_{\mathbf{B}^{\Phi}} \mathbf{L} =& \|\mathbf{U}(\mathbf{B}^{\Phi})^{\mathbf{T}} - k'\mathbf{S}^{\Phi}\|_2^2 + \lambda\|\mathbf{U}^{\phi^*} - \mathbf{B}^{\phi}\|_2^2 \\ & s.t.\ \mathbf{B}^{\Phi} \in \{-1, +1\}^{n\times k'}. \end{aligned} \tag{11}$$

It is not straightforward to update $\mathbf{B}^{\Phi}$. We first denote $\tilde{\mathbf{U}}^{\Phi} = \{\tilde{u}_i | i \in \Phi\} \in [-1, +1]^{n\times k'}$, where

$$\tilde{u}_i = \begin{cases} u_{i^*}, \text{if } i^* \in \phi^* \\ 0, \text{otherwise} \end{cases} \tag{12}$$

Note that a one-to-one mapping exists between i and i^* if $i^* \in \phi^*$, and the Eqn. (11) can be rewritten into follows:

$$\begin{aligned} \min_{\mathbf{B}^{\Phi}} \mathbf{L} =& \|\mathbf{U}(\mathbf{B}^{\Phi})^{\mathbf{T}} - k'\mathbf{S}^{\Phi}\|_2^2 + \lambda\|\tilde{\mathbf{U}}^{\Phi} - \mathbf{B}^{\Phi}\|_2^2 \\ =& \|\mathbf{B}^{\Phi}\mathbf{U}^{\mathbf{T}}\|_2^2 + tr(\mathbf{B}^{\Phi}\mathbf{V}^{\mathbf{T}}) + const \\ & s.t.\ \mathbf{B}^{\Phi} \in \{-1, +1\}^{n\times k'}. \end{aligned} \tag{13}$$

where $\mathbf{V} = -2k'(\mathbf{S}^{\Phi})^{\mathbf{T}}\mathbf{U} - 2\lambda\tilde{\mathbf{U}}^{\Phi}$. Then we adopt the discrete cyclic coordinate descent (DCC) algorithm proposed in (Shen et al. 2015) to optimize $\mathbf{B}^{\Phi}$ column by column. Finally, we can get the optimal solution of the above equation as follows:

$$\mathbf{B}_{*i}^{\Phi} = -sign(2\hat{\mathbf{B}}_i^{\Phi}\hat{\mathbf{U}}_i^{\mathbf{T}}\mathbf{U}_{*i} + \mathbf{V}_{*i}) \tag{14}$$

where $\mathbf{B}_{*i}^{\Phi}$ denotes as the ith column of $\mathbf{B}^{\Phi}$ and $\hat{\mathbf{B}}_i^{\Phi}$ represents the matrix of $\mathbf{B}^{\Phi}$ excluding $\mathbf{B}_{*i}^{\Phi}$. $\mathbf{V}_{*i}$ denotes the ith column of $\mathbf{V}$. $\mathbf{U}_{*i}$ denotes the ith column of $\mathbf{U}$ and $\hat{\mathbf{U}}_i$ denotes the matrix of $\mathbf{U}$ excluding $\mathbf{U}_{*i}$.

Experiments

Datasets

We conduct extensive experiments on three public benchmark image retrieval datasets: CIFAR-10 (Krizhevsky and Hinton 2009), NUS-WIDE (Chua et al. 2009) and ImageNet (Lin et al. 2014).

- **CIFAR-10** is a dataset containing 60,000 color images in 10 classes, and each class contains 6,000 images with a resolution of 32×32. Following (Lai et al. 2015), we randomly select 1,000 images (100 images per class) as the test query set, and 5,000 images (500 images per class) as the training set.
- **NUS-WIDE** is a large multi-label dataset containing 269,648 images across 81 classes. Following (Li, Wang, and Kang 2016), we select a subset of 195,834 images belonging to the 21 most frequent classes. To create query set, we randomly selected 2,100 images, ensuring that each class is represented by 100 images. For the training set, we randomly chose 10,500 images from the remaining images, ensuring that each class is represented by 500 images.

Datasets	#Original/#Incremental			
	S1	S2	S3	S4
CIFAR-10	4/6	6/4	7/3	8/2
NUS-WIDE	10/11	17/4	18/3	19/2
ImageNet	40/60	60/40	70/30	80/20

Table 1: Split details of three datasets.

- **ImageNet** is a single-label dataset containing over 1.2 million images from ILSVRC. Each image is associated with one of the 1,000 classes. As described in (Li, Wang, and Kang 2016), we randomly select 100 classes to constitute the retrieval set. The validation set is utilized to form the query set. Furthermore, we randomly select 10,000 images (100 images per class) for the training set.

Baselines

We choose some representative and also competitive deep hashing methods like ADSH (Jiang and Li 2018), HashNet (Cao et al. 2017), DSDH (Li et al. 2017), DPSH (Li, Wang, and Kang 2016) as our backbone methods. Note that for a fair comparison, we choose pairwise-label-based deep hashing methods as our baselines. In real applications, a similarity matrix is more likely to be generated than pointwise labels, especially for multi-label images. For shallow methods, we choose SH (Weiss, Torralba, and Fergus 2009), ITQ (Gong and Lazebnik 2011), LFH (Zhang et al. 2014) and SDH (Shen et al. 2015). We try our best to re-implement the previous methods and implement our CEDIH based on the corresponding backbone methods.

Incremental Learning Setting

We split the datasets into two parts, i.e., the original and incremental sets. Split details are shown in Table 1. For each dataset, we design four split settings separately. In single-label datasets CIFAR-10 and ImageNet, for example, "7/3" represents the images in the original set are from 7 classes while the incremental set includes images from the other 3 classes. In multi-label dataset NUS-WIDE, for example, "18/3" denotes the images in the original set are associated with at most 18 concepts, while the images in the incremental set are associated with at least one concept of the remaining 3 concepts. To explore the capability boundaries of our CEDIH, we further design three challenging settings (S1), in which the number of incremental classes is larger than that of the original classes.

Evaluation Metrics

We report the Mean Average Precision (mAP) and Precision-Recall curves (PR curves) to evaluate the retrieval performances. Regarding mAP results computation, for NUS-WIDE and ImageNet, the mAP results are calculated based on Top-5K and Top-1K returned samples, respectively. For CIFAR-10, the mAP results are obtained based on all returned images. Notably, we list the mAP results of CEDIH under different incremental learning settings. To verify the time efficiency of CEDIH, we further report the code expansion time.

Methods	CIFAR-10@ALL			NUS-WIDE@5000			ImageNet@1000		
	24 bits	32 bits	48 bits	24 bits	32 bits	48 bits	24 bits	32 bits	48 bits
SH	0.1912	0.1892	0.2044	0.5886	0.6402	0.6309	0.2421	0.2806	0.3235
ITQ	0.2215	0.2308	0.2386	0.7016	0.7186	0.7280	0.2732	0.3296	0.3751
LFH	0.4032	0.4047	0.4302	0.6903	0.7131	0.7321	0.2999	0.3738	0.4327
SDH	0.5209	0.5373	0.5311	0.7745	0.7932	0.7912	0.4790	0.5096	0.5429
DPSH	0.7274	0.7551	0.7510	0.8283	0.8360	0.8427	0.4006	0.4433	0.4622
CEDIH+DPSH	**0.7622**	**0.7701**	**0.7790**	**0.8479**	**0.8558**	**0.8614**	**0.4904**	**0.5423**	**0.5776**
HashNet	0.7561	0.7649	0.7726	0.8210	0.8278	0.8351	0.4099	0.4653	0.5311
CEDIH+HashNet	**0.7631**	**0.7857**	**0.7897**	**0.8476**	**0.8514**	**0.8618**	**0.5259**	**0.5734**	**0.6171**
DSDH	0.7441	0.7497	0.7530	0.8257	0.8316	0.8394	0.3967	0.4267	0.4718
CEDIH+DSDH	**0.7657**	**0.7667**	**0.7698**	**0.8466**	**0.8563**	**0.8644**	**0.4719**	**0.4966**	**0.5655**
ADSH	0.7716	0.7740	0.7787	0.8169	0.8280	0.8396	0.5790	0.6157	0.6653
CEDIH+ADSH	**0.7782**	**0.7811**	**0.7898**	**0.8328**	**0.8398**	**0.8481**	**0.6078**	**0.6413**	**0.6728**

Table 2: Comparison of mAP w.r.t different number of bits on three datasets. The best accuracy is shown in bold.

Methods	24 bits	32 bits	48 bits	64 bits
DIHN+ADSH	0.8228	0.8329	0.8403	0.8488
CEDIH+ADSH	**0.8328**	**0.8398**	**0.8481**	**0.8556**

Table 3: Comparison to the deep incremental hashing method DIHN on NUS-WIDE under the setting 3.

Setting	CIFAR-10	NUS-WIDE	ImageNet
-	0.7787	0.8396	0.6653
1	0.7799	**0.8554**	0.6278
2	**0.8013**	0.8474	0.6616
3	0.7898	0.8481	**0.6728**
4	0.7796	0.8414	0.6665

Table 4: Comparison of mAP w.r.t different split settings on three datasets. The backbone method is ADSH. The original length is 44 and the expanded length is 48.

Methods	c=8	c=12	c=16	c=20
ADSH	0.8424	0.8393	0.8433	0.8456
CEDIH+ADSH	0.8503	0.8468	0.8498	0.8512

Table 5: Comparison of mAP w.r.t different length of new hash codes on NUS-WIDE under the setting 3. The original length is 44.

Accuracy Comparison

Comparison to the state-of-the-art methods. Table 2 lists the mAP results of our CEDIH and the state-of-the-art deep hashing methods (SOTA) with different bits on three datasets. For the incremental learning setting, we adopt the setting 3 (S3). For CEDIH, we set $c = 4$, meaning that CEDIH with k' bits is expanded from the corresponding backbone methods with $k' - 4$ bits ($k' = 24, 32, 48$). In all cases, the retrieval performances of CEDIH are better than the corresponding backbone methods with the same code length. Besides, CEDIH with a better backbone method will achieve better retrieval performances, which is in line with expectations. The results also demonstrate that our method is compatible with a variety of deep hashing methods.

Comparison to the deep incremental hashing method. We further compare our method with the representative deep incremental hashing method DIHN (Wu et al. 2019), the results are listed in the Table 3. As DIHN cannot achieve code expansion, we directly train backbone method ADSH with the target length k' and adopt DIHN to incrementally learn new classes. For CEDIH, we still set $c = 4$. As shown in the table, our method can gain stable advantages, verifying the effectiveness of the simultaneous learning of new bits and the new classes.

Performance under different incremental learning settings. To fully verify the effectiveness of CEDIH, we list the mAP results of CEDIH under different settings on three datasets in Table 4. The first line lists the mAP results of the corresponding backbone method ADSH. In most cases, CEDIH can achieve advantages over ADSH on three datasets. However, for the setting 1 and 2 on ImageNet, we are inferior to the backbone method ADSH. We argue that the new bits are directly generated from the fixed original database codes, the semantic capacity of which is limited.

Performance with different length of new bits. Table 5 lists the mAP results of CEDIH with different length of new bits on ImageNet under the setting 3. The original code length is 44. As shown in the table, our method can still gain obvious advantages against the backbone method ADSH.

We attribute our advantages to the guidance of fixed original database codes and the asymmetric framework. On the one hand, the fixed original database codes can serve as the pre-defined target for both original and new images. On the other hand, as observed in (Neyshabur et al. 2013), the asymmetric framework can benefit the learning of hash codes. In CEDIH, the projection matrix for expanding original database codes and the new CNN model for learning new classes are optimized asymmetrically.

Code Expansion Time

Table 6 lists the code expansion time of CEDIH and the corresponding backbone methods with GPU on ImageNet under the setting 3. The original code length is 44 and the expanded code length is 48. For our CEDIH, the code expan-

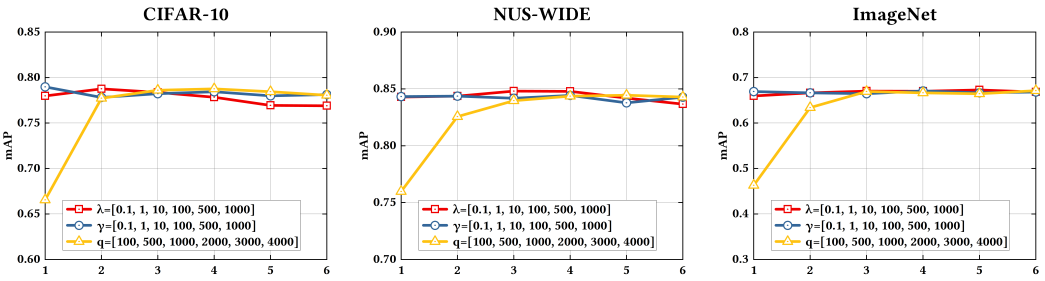

Figure 4: Parameter sensitivity on three datasets. The backbone model is ADSH and the code length is 48. Best viewed in color.

Methods	24 bits	32 bits	48 bits	64 bits
DPSH	153.31	153.59	154.06	154.18
DSDH	153.62	153.72	154.30	154.50
HashNet	152.75	153.31	153.63	154.50
ADSH	153.02	153.64	153.70	153.80
CEDIH+DPSH	46.21	46.36	46.49	46.62
CEDIH+DSDH	46.18	46.38	46.48	46.62
CEDIH+HashNet	46.22	46.25	46.31	46.41
CEDIH+ADSH	46.30	46.39	46.50	46.81

Table 6: Comparison of code expansion time on ImageNet (in seconds).

DEQ loss			Datasets		
T1	T2	T3	CIFAR-10	NUS-WIDE	ImageNet
			0.7804	0.8398	0.6659
✔	✔		0.7836	0.8398	0.6716
✔		✔	0.7830	0.8420	0.6705
	✔	✔	0.7823	0.8403	0.6690
✔	✔	✔	0.7898	0.8481	0.6728

Table 7: Ablation studies on the double-end quantization loss under the setting 3. The backbone method is ADSH, the original length is 44 and the expanded length is 48. T is short for Term.

sion time includes two parts: original database code expansion time and new database code generation time. For the corresponding backbone methods, the code expansion time only includes database code generation time. As shown in the table, our CEDIH is nearly four times faster than the corresponding backbone methods on GPU. Note that the original database code expansion time of our CEDIH can almost be ignored. Our time costs are mainly from the images from new classes, which are much less than the corresponding backbone methods. As the traditional deep hashing methods have to regenerate hash codes for all database images.

Parameter Sensitivity

To investigate the sensitivity of λ, γ, and q, we further conduct experiments under different values of λ, γ, and q on three datasets under the setting 3. The backbone method is ADSH, the original code length is 44 and the expanded code length is 48. The mAP results are illustrated in Figure 4. We tune λ in the range of $[0.1, 1000]$ by fixing $\{\gamma = 1, q = 2000\}$ for CIFAR-10, NUS-WIDE, and ImageNet. Similarly, we set $\{\lambda = 1, q = 2000\}$ for CIFAR-10, NUS-WIDE, and ImageNet when tuning γ. When tuning q, we set $\{\lambda = 1, \gamma = 1\}$ for CIFAR-10, NUS-WIDE, and ImageNet. In general, λ and γ have a wide range of $[0.1, 1000]$. For q, sampling more query images can achieve better retrieval performance at first. When q reaches a certain value, the mAP results will converge.

Ablation of Double-End Quantization Loss

To fully investigate the effectiveness of each term in the proposed double-end quantization loss (DEQ loss), we conduct experiments to evaluate the performances of CEDIH with different quantization terms. The results are listed in Table 7. Specifically, we report the mAP results of CEDIH without the proposed double-end quantization loss (the first line) or with one quantization term removed (the second line to the fifth line). As shown in the table, on the one hand, the proposed double-end quantization loss plays an important role in our method. For example, without the double-end quantization loss, the mAP result of CEDIH on CIFAR-10 drops to 0.7804 (1.2% decrease). On the other hand, each quantization term also more or less contributes to the final performance of CEDIH.

Conclusion

This paper introduces a unified deep hashing framework, called Code Expansion enabled Deep Incremental Hashing (CEDIH), for simultaneous new classes and expanded hash code learning. To the best of our knowledge, CEDIH is the first deep incremental hashing method equipped with fast code expansion ability. Benefiting from the asymmetric design of CEDIH and the proposed pairwise-label-based incremental similarity-preserving loss, the similarities between new and original images can be incrementally preserved and embedded into the expanded original database codes with the original ones unchanged. Furthermore, we comprehensively consider the quantization errors introduced by the new and original query images, and we carefully design a double-end quantization loss, thus further boosting retrieval performance. Extensive experiments demonstrate that our CEDIH can achieve fast code expansion while incrementally learning new classes with no loss of accuracy.

Acknowledgments

This work was supported by the National Key R&D Program of China under Grant 2022YFB3103500, the National Natural Science Foundation of China under Grants 62106258, 62006242 and 62202459, and the China Postdoctoral Science Foundation under Grant 2022M713348 and 2022TQ0363, and Young Elite Scientists Sponsorship Program by BAST (NO.BYESS2023304).

References

Cakir, F.; He, K.; Bargal, S. A.; and Sclaroff, S. 2019. Hashing with mutual information. *IEEE Trans. on Pattern Analalysis and Machine Intelligence (TPAMI)*, 41(10): 2424–2437.

Cakir, F.; He, K.; and Sclaroff, S. 2018. Hashing with binary matrix pursuit. In *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 332–348.

Cao, Z.; Long, M.; Wang, J.; and Yu, P. S. 2017. Hashnet: Deep learning to hash by continuation. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 5608–5617.

Chen, Y.; Lai, Z.; Ding, Y.; Lin, K.; and Wong, W. K. 2019. Deep Supervised Hashing With Anchor Graph. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 9795–9803.

Chua, T.-S.; Tang, J.; Hong, R.; Li, H.; Luo, Z.; and Zheng, Y. 2009. NUS-WIDE: a real-world web image database from National University of Singapore. In *Proc. of the ACM Intl. Conf. on Multimedia Retrieval (ICMR)*, 1–9.

Cui, H.; Zhu, L.; Li, J.; Yang, Y.; and Nie, L. 2019. Scalable Deep Hashing for Large-Scale Social Image Retrieval. *IEEE Trans. on Image Processing (TIP)*, 1271–1284.

Cui, H.; Zhu, L.; Li, J.; Zhang, Z.; and Guan, W. 2022. Webly Supervised Image Hashing with Lightweight Semantic Transfer Network. In *Proc. of the ACM Intl. Conf. on Multimedia (ACM MM)*, 3451–3460.

Deng, C.; Chen, Z.; Liu, X.; Gao, X.; and Tao, D. 2018. Triplet-based deep hashing network for cross-modal retrieval. *IEEE Trans. on Image Processing (TIP)*, 27(8): 3893–3903.

Doan, K. D.; Yang, P.; and Li, P. 2022. One loss for quantization: Deep hashing with discrete wasserstein distributional matching. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 9447–9457.

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 *Proc. of the Intl. Conf. on Artificial Intelligence (IJCAI)*, 825–831.

Gionis, A.; Indyk, P.; and Motwani, R. 1999. Similarity Search in High Dimensions via Hashing. In *Proc. of the International Conference on Very Large Data Bases (VLDB)*, 518–529.

Gong, Y.; and Lazebnik, S. 2011. Iterative quantization: A procrustean approach to learning binary codes. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 817–824.

Gu, W.; Gu, X.; Gu, J.; Li, B.; Xiong, Z.; and Wang, W. 2019. Adversary guided asymmetric hashing for cross-modal retrieval. In *Proc. of the ACM Intl. Conf. on Multimedia Retrieval (ICMR)*, 159–167.

Hoe, J. T.; Ng, K. W.; Zhang, T.; Chan, C. S.; Song, Y.-Z.; and Xiang, T. 2021. One loss for all: Deep hashing with a single cosine similarity based learning objective. *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*, 34: 24286–24298.

Jiang, Q.-Y.; and Li, W.-J. 2018. Asymmetric deep supervised hashing. In *Proc. of the Conf. Artif. Intell. (AAAI)*, 3342–3349.

Jin, S.; Yao, H.; Sun, X.; Zhou, S.; Zhang, L.; and Hua, X. 2020. Deep saliency hashing for fine-grained retrieval. *IEEE Trans. on Image Processing (TIP)*, 29: 5336–5351.

Krizhevsky, A.; and Hinton, G. 2009. Learning multiple layers of features from tiny images. Technical report.

Lai, H.; Pan, Y.; Liu, Y.; and Yan, S. 2015. Simultaneous feature learning and hash coding with deep neural networks. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 3270–3278.

Li, Q.; Sun, Z.; He, R.; and Tan, T. 2017. Deep supervised discrete hashing. In *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*, 2482–2491.

Li, W.-J.; Wang, S.; and Kang, W.-C. 2016. Feature learning based deep supervised hashing with pairwise labels. In *Proc. of the Intl. Conf. on Artificial Intelligence (IJCAI)*, 1711–1717.

Lin, T. Y.; Maire, M.; Belongie, S.; Hays, J.; and Zitnick, C. L. 2014. Microsoft COCO: Common Objects in Context. In *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 740–755.

Liu, B.; Cao, Y.; Long, M.; Wang, J.; and Wang, J. 2018a. Deep triplet quantization. In *Proc. of the ACM Intl. Conf. on Multimedia (ACM MM)*, 755–763.

Liu, H.; Wang, R.; Shan, S.; and Chen, X. 2016. Deep supervised hashing for fast image retrieval. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2064–2072.

Liu, X.; Nie, X.; Zeng, W.; Cui, C.; Zhu, L.; and Yin, Y. 2018b. Fast discrete cross-modal hashing with regressing from semantic labels. In *Proc. of the ACM Intl. Conf. on Multimedia (ACM MM)*, 1662–1669.

Liu, X.; Nie, X.; Zhou, Q.; Xi, X.; Zhu, L.; Yin, Y.; et al. 2019. Supervised Short-Length Hashing. In *Proc. of the Intl. Conf. on Artificial Intelligence (IJCAI)*, 3031–3037.

Luo, X.; Zhang, P. F.; Huang, Z.; Nie, L.; and Xu, X. S. 2019. Discrete Hashing With Multiple Supervision. *IEEE Trans. on Image Processing (TIP)*, 2962–2975.

Luo, X.; Zhang, P.-F.; Wu, Y.; Chen, Z.-D.; Huang, H.-J.; and Xu, X.-S. 2018. Asymmetric discrete cross-modal hashing. In *Proc. of the ACM Intl. Conf. on Multimedia Retrieval (ICMR)*, 204–212.

Mandal, D.; Annadani, Y.; and Biswas, S. 2019. Growbit: Incremental hashing for cross-modal retrieval. In *Proc. of the Asi. Conf. Comput. Vis. (ACCV)*, 305–321.

Mandal, D.; and Biswas, S. 2020. A novel incremental cross-modal hashing approach. *arXiv preprint*.

Neyshabur, B.; Srebro, N.; Salakhutdinov, R. R.; Makarychev, Y.; and Yadollahpour, P. 2013. The power of asymmetry in binary hashing. In *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*, 2823–2831.

Shen, F.; Gao, X.; Liu, L.; Yang, Y.; and Shen, H. T. 2017. Deep Asymmetric Pairwise Hashing. In *Proc. of the ACM Intl. Conf. on Multimedia (ACM MM)*, 1522–1530.

Shen, F.; Shen, C.; Liu, W.; and Tao Shen, H. 2015. Supervised discrete hashing. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 37–45.

Shen, Y.; Qin, J.; Chen, J.; Liu, L.; Zhu, F.; and Shen, Z. 2019. Embarrassingly simple binary representation learning.

Song, W.; Gao, Z.; Dian, R.; Ghamisi, P.; Zhang, Y.; and Benediktsson, J. A. 2022. Asymmetric hash code learning for remote sensing image retrieval. *IEEE Trans. on Geoscience and Remote Sensing (TGRS)*, 60: 1–14.

Tang, J.; Lin, J.; Li, Z.; and Yang, J. 2018. Discriminative Deep Quantization Hashing for Face Image Retrieval. *IEEE Trans. on Neural Networks and Learning Systems (TNNLS)*, 6154–6162.

Tian, X.; Ng, W. W.; and Xu, H. 2023. Deep Incremental Hashing for Semantic Image Retrieval with Concept Drift. *IEEE Trans. on Big Data (TBD)*.

Tu, J.; Liu, X.; Lin, Z.; Hong, R.; and Wang, M. 2022. Differentiable Cross-modal Hashing via Multimodal Transformers. In *Proc. of the ACM Intl. Conf. on Multimedia (ACM MM)*, 453–461.

Weiss, Y.; Torralba, A.; and Fergus, R. 2009. Spectral hashing. In *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*, 1753–1760.

Wu, D.; Dai, Q.; Li, B.; and Wang, W. 2023. Deep Uncoupled Discrete Hashing via Similarity Matrix Decomposition. *ACM Trans. on Multimedia Computing, Communications and Applications (TOMM)*.

Wu, D.; Dai, Q.; Liu, J.; Li, B.; and Wang, W. 2019. Deep Incremental Hashing Network for Efficient Image Retrieval. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 9069–9077.

Wu, D.; Lin, Z.; Li, B.; Ye, M.; and Wang, W. 2017. Deep Supervised Hashing for Multi-Label and Large-Scale Image Retrieval. In *Proc. of the ACM Intl. Conf. on Multimedia Retrieval (ICMR)*, 150–158.

Wu, D.; Su, Q.; Li, B.; and Wang, W. 2022. Efficient Hash Code Expansion by Recycling Old Bits. In *Proc. of the ACM Intl. Conf. on Multimedia (ACM MM)*, 572–580.

Xia, R.; Pan, Y.; Lai, H.; Liu, C.; and Yan, S. 2014. Supervised Hashing for Image Retrieval via Image Representation Learning. In *Proc. of the Conf. Artif. Intell. (AAAI)*, 2156–2162.

Yang, H.-F.; Lin, K.; and Chen, C.-S. 2018. Supervised Learning of Semantics-Preserving Hash via Deep Convolutional Neural Networks. *IEEE Trans. on Pattern Analalysis and Machine Intelligence (TPAMI)*, 40(2): 437.

Yao, T.; Long, F.; Mei, T.; and Rui, Y. 2016. Deep semantic-preserving and ranking-based hashing for image retrieval. In *Proc. of the Intl. Conf. on Artificial Intelligence (IJCAI)*.

Yuan, L.; Wang, T.; Zhang, X.; Tay, F. E.; Jie, Z.; Liu, W.; and Feng, J. 2020. Central similarity quantization for efficient image and video retrieval. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 3083–3092.

Zhang, D.; Wu, X.-J.; Xu, T.; and Yin, H. 2021a. Dah: discrete asymmetric hashing for efficient cross-media retrieval. *IEEE Trans. on Knowledge and Data Engineering (TKDE)*.

Zhang, H.; Liu, L.; Long, Y.; and Shao, L. 2018. Unsupervised Deep Hashing With Pseudo Labels for Scalable Image Retrieval. *IEEE Trans. on Image Processing (TIP)*, 1626–1638.

Zhang, P.; Zhang, W.; Li, W.-J.; and Guo, M. 2014. Supervised hashing with latent factor models. In *Proc. of the ACM Intl. Conf. on Research and Development in Information Retrieval (SIGIR)*, 173–182.

Zhang, P.-F.; Li, C.-X.; Liu, M.-Y.; Nie, L.; and Xu, X.-S. 2017. Semi-relaxation supervised hashing for cross-modal retrieval. In *Proc. of the ACM Intl. Conf. on Multimedia (ACM MM)*, 1762–1770.

Zhang, P.-F.; Li, Y.; Huang, Z.; and Xu, X.-S. 2021b. Aggregation-based graph convolutional hashing for unsupervised cross-modal retrieval. *IEEE Trans. on Multimedia (TMM)*, 24: 466–479.

Zhang, W.; Wu, D.; Li, B.; Gu, X.; Wang, W.; and Meng, D. 2019. Fast and multilevel semantic-preserving discrete hashing. In *Proc. of Brit. Mach. Vis. Conf. (BMVC)*.

Zhang, W.; Wu, D.; Zhou, Y.; Li, B.; Wang, W.; and Meng, D. 2020. Deep unsupervised hybrid-similarity hadamard hashing. In *Proc. of the ACM Intl. Conf. on Multimedia (ACM MM)*, 3274–3282.

Zhang, W.; Wu, D.; Zhou, Y.; Li, B.; Wang, W.; and Meng, D. 2021c. Binary neural network hashing for image retrieval. In *Proc. of the ACM Intl. Conf. on Research and Development in Information Retrieval (SIGIR)*, 1318–1327.

Zhao, F.; Huang, Y.; Wang, L.; and Tan, T. 2015. Deep semantic ranking based hashing for multi-label image retrieval. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 1556–1564.

Zhao, S.; Wu, D.; Zhang, W.; Zhou, Y.; Li, B.; and Wang, W. 2020. Asymmetric Deep Hashing for Efficient Hash Code Compression. In *Proc. of the ACM Intl. Conf. on Multimedia (ACM MM)*, 763–771.

Zhao, S.; Wu, D.; Zhou, Y.; Li, B.; and Wang, W. 2021. Rescuing deep hashing from dead bits problem. In *Proc. of the Intl. Conf. on Artificial Intelligence (IJCAI)*.

Zhuang, B.; Lin, G.; Shen, C.; and Reid, I. 2016. Fast Training of Triplet-Based Deep Binary Embedding Networks. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 5955–5964.