Deep Graph Online Hashing for Multi-Label Image Retrieval

Yuan Cao[1], Xiangru Chen[1], Zifan Liu*[1], Wenzhe Jia[1], Fanlei Meng[1], Jie Gui*[2,3,4]

[1] School of Computer Science and Technology, Ocean University of China, Qingdao, China
[2] School of Cyber Science and Engineering, Southeast University, Nanjing, China
[3] Engineering Research Center of Blockchain Application, Supervision And Management (Southeast University), Ministry of Education, Nanjing, China
[4] Purple Mountain Laboratories, Nanjing, China
cy8661@ouc.edu.cn, {chenxiangru, lzf9793, jiawenzhe, mengfanlei}@stu.ouc.edu.cn, guijie@seu.edu.cn

Abstract

Online hashing has attracted much research attention for large-scale image retrieval in a streaming way. The main challenge lies in keeping balance between high retrieval accuracy and low training time. Existing online hashing methods almost rely on shallow models rather than deep networks due to high training costs, because it is unacceptable to update hash functions on an order of hours. In addition, the multi-label supervision information is not fully utilized to guide the hash learning process and the affinity matrix is always fixed once constructed. In this paper, we propose a novel Deep Graph Online Hashing (DGOH) method, which for the first time introduces inductive graph neural networks (GNNs) to realize deep online hashing with acceptable training costs on an order of seconds. Furthermore, we mine the multi-label information of the images by constructing a label network and learn label-wise weights dynamically to help to update the affinity matrix. In addition, we provide a strategy to obtain examples from the old data to solve the catastrophic forgetting problem. An integrated objective function is designed to train the entire architecture. Extensive experiments on two common benchmarks demonstrate that the proposed method achieves up to 13.3% accuracy gains over state-of-the-art baselines and shows competitive performance on training time.

Code — https://github.com/caoyuan57/DGOH

Introduction

With the explosive growth of image data, finding efficient retrieval technology for them has become an urgent problem. Hashing (Luo et al. 2023; Wang et al. 2023) is one of the most popular and attractive strategies due to its high search efficiency and low storage cost. Most existing hashing methods aim at learning hash functions from a fixed collection of data (Xia et al. 2014; Gui et al. 2017; Wang et al. 2021). However, image data often accumulates in a streaming way in practical applications. As a result, offline hashing cannot adapt to this scenario because the hash functions fail to be updated dynamically and efficiently.

To address this issue, online hashing (Xie, Shen, and Zhu 2016; Lu et al. 2019; Fang, Zhang, and Liu 2021; Wu et al. 2022; Liu et al. 2022; Zhang, Wu, and Chen 2023) has recently received extensive attention by updating hash functions instantly in consideration of new streaming data with low computational complexity. Online hashing methods can be categorized into unsupervised ones (Leng et al. 2015; Chen, King, and Lyu 2017) and supervised ones (Lin et al. 2020a; Weng and Zhu 2020; Zhan et al. 2021; Jin et al. 2021) depending on whether the labels are utilized in the training process. Unsupervised online hashing is mostly designed on the idea of data sketch to help to analyze the similarities among data in consideration of the inherent attributes (i.e. distribution and variance) of data. Online sketch hashing (OSH) (Leng et al. 2015) uses a smaller sketch matrix to preserve the main property of data but with a significantly smaller size. The hash functions are learnt by maximizing the variance for each hash bit in the sketch matrix. Chen (Chen, King, and Lyu 2017) present a faster online sketching hashing (FROSH) algorithm to improve the efficiency of OSH by developing a faster frequent direction algorithm via utilizing the subsampled randomized Hadamard transform. In general, unsupervised online hashing is less accurate than supervised online hashing because the labels are is not leveraged for training.

As the first success in supervised online hashing, online kernel hashing (OKH) (Huang, Yang, and Zheng 2013) derives and optimizes a structured hash model in a passive-aggressive way. In order to solve the "data imbalance" and "relaxation" problems, Lin (Lin et al. 2019) introduce the balance factor into the online discrete hashing (BSODH) model to adjust the similar and dissimilar data pairs in the training process. Lin (Lin et al. 2020b) propose a Hadamard matrix guided online hashing (HMOH) and adopts LSH (Datar et al. 2004) to reduce the length of Hadamard codes in accordance with the hash bits to guarantee the semantic similarity. Since it is inefficient to update the hash codes of the entire database with the latest hash functions when new streaming data arrives, Jia (Jia et al. 2023) propose a fast online hashing (FOH) method by building a query pool and presenting a neighbor-preserving algorithm to record the nearest neighbors of the central points.

However, two core challenges have been ignored. On the one hand, these methods all rely on shallow models, which limits the retrieval accuracy, creating a bottleneck. Online deep hashing (ODHUC) (Xie et al. 2022) attempts to in-

*Corresponding author
Copyright © 2025, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

troduce deep networks into online hashing. Nevertheless, ODHUC only trains the deep network with batch data each time. The training time is almost the same as that of general deep networks. Hence, ODHUC is actually not considered as a pure online hashing method. On the other hand, the multi-label information is not fully mined and the affinity matrix is fixed once constructed.

In this paper, we propose a novel online hashing method, termed Deep Graph Online Hashing (DGOH) based on inductive GNNs (Hamilton, Ying, and Leskovec 2017) which takes the nodes and edges of a graph as input to learn node representations from existing and new dynamic graphs. GNNs show superior accuracy performance compared with shallow methods, because they have multiple layers and more parameters that allow them to learn increasingly high-level semantic representations of data. Furthermore, in GNNs, graph data is often sparse, which leads to more efficient computations during training (hundreds of seconds), as only the connected nodes need to be considered. To be specific, we construct a bilayer architecture composed of two inductive GNNs: feature network and label network. In the feature network, the image features are first extracted from the vision GNN backbone (Han et al. 2022), and then fed into the feature graph which is constructed by the affinity matrix. The binary codes of the image features are generated dynamically as the streaming data accumulates. In the label network, the labels are first embedded by the Word2Vec backbone (Mikolov et al. 2013), and then fed into the label graph which is constructed by the multi-label images. In addition, the label-wise weights are also updated simultaneously with the label features being generated. Therefore, the affinity matrix and feature graph can both be updated dynamically according to the latest label-wise weights. Since the label graph is constructed iteratively, new labels are also easy to be identified. In order to solve the catastrophic forgetting problem, we provide a strategy to select dissimilar old examples rather than random selection to preserve similarities between the existing data and new arrivals. Moreover, we design an integrated objective function to train the feature and label networks iteratively and dynamically.

The main contributions are summarized as: 1) We realize deep graph online hashing by introducing inductive GNNs into our model to keep balance between retrieval accuracy and training time. 2) We construct a label network to mine multi-label information and learn the label-wise weights dynamically to update the affinity matrix. 3) Extensive experiments on two widely used image datasets demonstrate that the proposed DGOH model achieves dramatic accuracy gains and competitive results in training time compared with state-of-the-art baselines for the image retrieval task.

The Proposed Method

The General Framework

The general framework of the proposed DGOH method is shown in Figure 1. The bilayer architecture consists of a feature network and a label network. In the former, we construct the feature graph by regarding the image features extracted from vision GNN as the nodes and generating the edges according to the affinity matrix. In the latter, we construct the label graph by regarding the label features produced by word2vector as the nodes and generating the edges in consideration of the multi-label images. The objective function mainly containing similarity preserving loss, label classification loss and quantization loss is used for optimizing all the parameters in the architecture.

Notations and Problem Definition

Suppose that the images arrive in a stream, let $\vec{\boldsymbol{X}}^t = \{\vec{\boldsymbol{x}}_i^t\}_{i=1}^{n^t} \in R^{n^t \times d_i}$ denote the new streaming image examples, where n^t denotes the batch size in stage t and d_i denotes the dimensionality of the image features encoded by vision GNN (Han et al. 2022). Let $\vec{\boldsymbol{L}}^t = \{\vec{\boldsymbol{l}}_i^t\}_{i=1}^{n^t} \in \{0,1\}^{n^t \times c}$ denote their corresponding labels, where c denotes the total number of categories. We denote $\vec{\boldsymbol{Y}}^t = \{\vec{\boldsymbol{y}}_i^t\}_{i=1}^{c_t} \in R^{c_t \times d_l}$ as the label features, where c_t denotes the number of labels accompanied by $\vec{\boldsymbol{X}}^t$ in stage t and d_l denotes the dimensionality of the label features embedded by word2vector (Mikolov et al. 2013). The previous image collections before state t are denoted as $\tilde{\boldsymbol{X}}^t = \{\tilde{\boldsymbol{x}}_i^t\}_{i=1}^{m^t} \in R^{m^t \times d}$ and accompanied by their labels $\tilde{\boldsymbol{L}}^t = \{\tilde{\boldsymbol{l}}_i^t\}_{i=1}^{m^t} \in \{0,1\}^{m^t \times c}$, where $m^t = n^1 + n^2 + ... + n^{t-1}$ denotes the number of previous image collections before stage t. The goal of online hashing is to encode the new streaming image examples $\vec{\boldsymbol{X}}^t$ into k-bit hash codes $\vec{\boldsymbol{B}}^t = \{\vec{\boldsymbol{b}}_i^t\}_{i=1}^{n^t} \in \{-1,1\}^{n^t \times k}$ to preserve similarities among the images. Similarly, $\tilde{\boldsymbol{B}}^t = \{\tilde{\boldsymbol{b}}_i^t\}_{i=1}^{m^t} \in \{-1,1\}^{m^t \times k}$ denotes the hash codes of the previous image collections. We define the hash function as

$$\vec{\boldsymbol{B}}^t = \mathrm{sgn}(F(\vec{\boldsymbol{X}}^t; \theta^t)), \tag{1}$$

where θ^t denotes the hash learning network parameters in stage t and the sign function $\mathrm{sgn}(x)$ is defined as

$$\mathrm{sgn}(x) = \begin{cases} 1, & x > 0, \\ -1, & otherwise. \end{cases} \tag{2}$$

In addition, $\|\cdot\|_F$ denotes the ℓ_F-norm of a vector or matrix.

Bilayer Architecture based Hashing

Feature Network The feature network is the core module of the overall architecture. The main target is to learn compact and discriminative hash codes for the streaming images. When a new batch of image examples arrive, we first construct a feature graph $< \mathcal{G}_\mathcal{F} = \mathcal{V}_\mathcal{F}, \mathcal{E}_\mathcal{F} >$, where $\mathcal{V}_\mathcal{F}$ is defined by the image features $\vec{\boldsymbol{X}}^t$ and $\mathcal{E}_\mathcal{F}$ is defined by the similarity relationships among the images, i.e.

$$\mathcal{V}_\mathcal{F} = \left\{ \boldsymbol{v}_{fi} = \vec{\boldsymbol{x}}_i^t \mid \vec{\boldsymbol{x}}_i^t \in \vec{\boldsymbol{X}}^t \right\}, \tag{3}$$

$$\mathcal{E}_\mathcal{F} = \left\{ (\boldsymbol{v}_{fi}, \boldsymbol{v}_{fj}) \mid S_{i,j} \geq \lambda \right\}, \tag{4}$$

where λ denotes the threshold hyperparameter and $S_{i,j}$ denotes the similarity between the i-th example and j-th example in the affinity matrix which is defined in the label network.

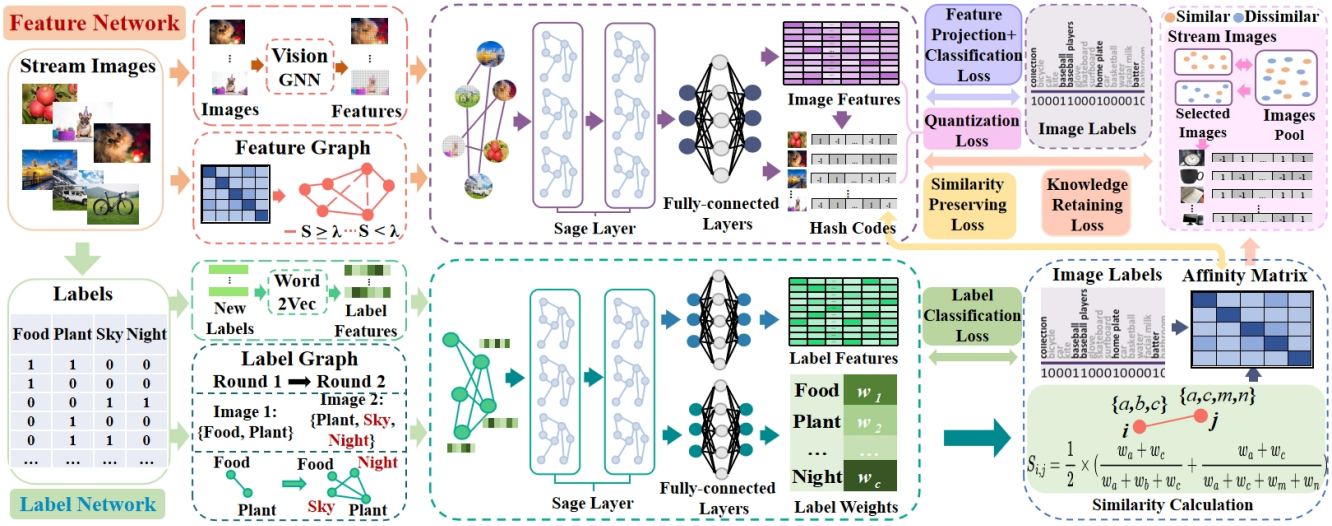

Figure 1: The general framework of the proposed Deep Graph Online Hashing method. The bilayer architecture is composed of a feature network and a label network. The feature network outputs the image features and corresponding hash codes. The label network outputs the label features and label-wise weights. The loss functions are used for training.

We then feed the feature graph $\mathcal{G}_{\mathcal{F}}$ into the two-layer inductive GNN (Han et al. 2022) to obtain the image feature matrix $\vec{\boldsymbol{X}}^{t*} \in R^{n_t \times d}$ which contains more information than $\vec{\boldsymbol{X}}^{t}$ because the features of adjacent nodes are aggregated, where d denotes the dimensionality of the GNN output layer. Finally, a fully-connected layer is added to each node to obtain the streaming images' hash codes $\vec{\boldsymbol{B}}^{t}$ in stage t:

$$\vec{\boldsymbol{B}}^{t} = \operatorname{sgn}(\sigma(\vec{\boldsymbol{X}}^{t*}\boldsymbol{W}_f)), \tag{5}$$

where σ denotes the activation function and $\boldsymbol{W}_f$ denotes the weights of the fully-connected layer in the feature network.

Label Network In the label network, we take the correlations among the labels into account aiming at learning label-wise weights to update the affinity matrix dynamically. Given the streaming images' labels $\vec{\boldsymbol{L}}^{t}$ and label features $\vec{\boldsymbol{Y}}^{t}$, we first construct a label graph $< \mathcal{G}_{\mathcal{L}} = \mathcal{V}_{\mathcal{L}}, \mathcal{E}_{\mathcal{L}} >$, where $\mathcal{V}_{\mathcal{L}}$ is defined by the label features $\vec{\boldsymbol{Y}}^{t}$ and $\mathcal{E}_{\mathcal{L}}$ is defined according to the streaming images' labels, i.e.

$$\mathcal{V}_{\mathcal{L}} = \left\{ \boldsymbol{v}_{li} = \vec{\boldsymbol{y}}_i^t \mid \vec{\boldsymbol{y}}_i^t \in \vec{\boldsymbol{Y}}^{t} \right\}, \tag{6}$$

$$\mathcal{E}_{\mathcal{L}} = \left\{ (\boldsymbol{v}_{li}, \boldsymbol{v}_{lj}) \mid \vec{\boldsymbol{L}}_{:,i}^{\mathrm{T}} \vec{\boldsymbol{L}}_{:,j} \neq 0 \right\}, \tag{7}$$

where $\vec{\boldsymbol{L}}_{:,i}$ and $\vec{\boldsymbol{L}}_{:,j}$ denote the i-th and j-th columns of $\vec{\boldsymbol{L}}$, respectively. We explain that if two labels ever appear in the same image, the edge between them exists. Hence, it relies on the multi-label information to construct the label graph, which explains that the proposed method is suitable for multi-label retrieval.

We then feed the label graph $\mathcal{G}_{\mathcal{L}}$ into the two-layer inductive GNN (Han et al. 2022) to obtain the label features $\vec{\boldsymbol{Y}}^{t*} \in R^{c_t \times d}$. Note that after the label graph is initialized by (6) and (7), the edges are added dynamically depending on the streaming images and the nodes' features are updated iteratively based on the aggregation operations in inductive GNN. Hence, new labels can be identified easily with the label network. We consider all the c labels together to transform the label features into $\vec{\boldsymbol{Y}}^{t*} \in R^{c \times d}$ for simplicity. Next, a fully-connected layer is added to each node for learning the label-wise weights $\boldsymbol{\omega}^{t}$ in stage t as

$$\boldsymbol{\omega}^{t} = N(\sigma(\vec{\boldsymbol{Y}}^{t*}\boldsymbol{W}_l)), \tag{8}$$

where N denotes the normalization to $[0, 1]$. Since all the labels are updated iteratively, we use the form of $\boldsymbol{\omega} = \{\omega_1, ..., \omega_c\}$ to denote all the label-wise weights for clarity which are in correspondence to the importance of the categories.

Finally, we calculate the affinity matrix $\vec{\boldsymbol{S}}^{t}$ based on the images' labels $\vec{\boldsymbol{L}}^{t}$ and label-wise weights $\boldsymbol{\omega}$ to evaluate the similarity between two images in stage t. We define

$$P = \{r \mid \vec{l}_{i,r}^{t} = 1\}, \tag{9}$$

$$Q = \{r \mid \vec{l}_{j,r}^{t} = 1\}, \tag{10}$$

$$O = P \cap Q, \tag{11}$$

where P and Q denotes the sets of label indices of the i-th and j-th images, respectively. Accordingly, O denotes the set of common label indices of the two images. Therefore, the similarity between the i-th and j-th images $\vec{S}_{i,j}^{t}$ in stage t is computed as

$$\vec{S}_{i,j}^{t} = \frac{\sum_{k \in O} \omega_k}{\sum_{i \in P} \omega_i + \sum_{j \in Q} \omega_j}. \tag{12}$$

We utilize the label-wise weights $\boldsymbol{\omega}$ to define fine-grained similarities. In other words, if a-th category is more important than b-th category, a-th category's weight contributes

more than b-th category's weight to the final similarity between two images.

Essential Loss Functions Since the hash learning process is crucial for the constructions of the various losses, we review that the hash function is defined as

$$\vec{\boldsymbol{B}}^t = \mathrm{sgn}(F(\vec{\boldsymbol{X}}^t; \theta^t)), \tag{13}$$

where $\vec{\boldsymbol{X}}^t$ denotes the new streaming image examples in stage t and $\vec{\boldsymbol{B}}^t$ denotes corresponding binary codes. In this part, we introduce the loss functions that are essential to train the whole architecture and obtain binary hash codes.

A. Similarity Preserving Loss

In order to preserve similarities among the new streaming images, we construct the similarity preserving loss $\mathcal{L}_s$ by utilizing the affinity matrix $\boldsymbol{S}^t$. In other words, if two image examples are similar, their binary codes should also be similar, which is formulated as an inner product loss:

$$\mathcal{L}_s = \left\| \vec{\boldsymbol{B}}^t \vec{\boldsymbol{B}}^{t^{\mathrm{T}}} - k\vec{\boldsymbol{S}}^t \right\|_F^2. \tag{14}$$

Note that the optimization problems associated with this loss function are NP-hard due to discrete constraints. Therefore, we approximate the discrete $\mathrm{sgn}(\cdot)$ function with the continuous $\tanh(\cdot)$ function and modify (14) as

$$\mathcal{L}_s = \left\| \tanh(F(\vec{\boldsymbol{X}}^t; \theta^t) \vec{\boldsymbol{B}}^{t^{\mathrm{T}}}) - k\vec{\boldsymbol{S}}^t \right\|_F^2, \tag{15}$$

where θ^t denotes the hash learning network parameters in stage t which is defined before.

B. Label Classification Loss

In order to train the label network, we construct the label classification loss to approximate the label features of a streaming image example and its labels. The most crucial step is to compute the mean of the label features $\vec{\boldsymbol{y}}_i^{t*}$ accompanied by a streaming image example $\vec{\boldsymbol{x}}_i$ in stage t:

$$U = \{r \mid \vec{l}_r^t = 1\}, \tag{16}$$

$$\vec{\boldsymbol{y}}_i^{t*} = \frac{1}{|U|} \sum_{i \in U} \vec{\boldsymbol{Y}}_{i,:}^{t*} \boldsymbol{W}_c, \tag{17}$$

where $\vec{\boldsymbol{Y}}^{t*} \in R^{c \times d}$ denotes the label features defined before and $\boldsymbol{W}_c \in R^{d \times c}$ denotes the label projection matrix.

Hence, the label classification loss $\mathcal{L}_c$ is formulated as the cross-entropy loss between the label probabilities and the labels:

$$\begin{aligned} \mathcal{L}_c &= -\frac{1}{n_t c} \sum_{i=1}^{n_t} \sum_{j=1}^{c} \vec{L}_{i,j} \log P_{i,j}^t \\ &= -\frac{1}{n_t c} \sum_{i=1}^{n_t} \sum_{j=1}^{c} \vec{L}_{i,j} \log \frac{e^{\vec{y}_{i,j}^{t*}}}{\sum_{j=1}^{c} e^{\vec{y}_{i,j}^{t*}}} \\ &= -\frac{1}{n_t c} \sum_{i=1}^{n_t} \sum_{j=c}^{c} \vec{L}_{i,j} \log \frac{e^{F_{i,j}(\vec{y}_i^t; \xi^t)}}{\sum_{j=1}^{c} e^{F_{i,j}(\vec{y}_i^t; \xi^t)}}, \end{aligned} \tag{18}$$

where ξ denotes the label feature learning network parameters in stage t.

C. Quantization Loss

The quantization loss is designed to minimize the error between the continuous values and discrete binary codes:

$$\mathcal{L}_q = \left\| F\left(\vec{\boldsymbol{X}}^t; \theta^t\right) - \vec{\mathbf{B}}^t \right\|_F^2. \tag{19}$$

We integrate (15), (18) and (19) to get the essential loss function $\mathcal{L}_e$:

$$\mathcal{L}_e = \alpha \mathcal{L}_s + \mathcal{L}_c + \mu \mathcal{L}_q, \tag{20}$$

where α and μ are the hyperparameters for balance.

Additional Loss Functions In this part, we introduce the loss functions that can further improve the accuracy.

A. Knowledge Retaining Loss

In order to alleviate the catastrophic forgetting problem caused by constantly updating the online model with new streaming image examples, we provide a novel strategy to sample old images from the previous image collections. The strategy is based on the idea that the old images which are as dissimilar as possible to the newly selected images.

We first calculate the indices Z of the labels accompanied by the new streaming images as

$$\vec{\boldsymbol{l}}^{t*} = \sum_{i=1}^{n^t} \vec{\boldsymbol{l}}_i^t, \tag{21}$$

$$Z = \{r \mid \vec{l}_r^{t*} \geq 0\}. \tag{22}$$

Then, we compute the indices of the sampled old images as

$$\boldsymbol{a} = \mathrm{sort}(\sum_{j \in Z} \tilde{\boldsymbol{L}}_{:,j}^t, m^{t*}), \tag{23}$$

where m^{t*} denotes the number of the sampled old images and $\mathrm{sort}(x, n)$ denotes an ascending sorting function to return a column vector of indices corresponding to the n smallest values in x. Finally, the hash codes $\tilde{\boldsymbol{B}}^{t*}$ of the sampled old images are obtained as

$$\tilde{\boldsymbol{B}}^{t*} = \tilde{\boldsymbol{B}}_{\boldsymbol{a},:}^t. \tag{24}$$

We explain that this strategy works because the dissimilarity correlations among the examples can be identified better, resulting in better similarity preservation in the loss. The performance is also verified by the experiments.

Therefore, the knowledge retaining loss $\mathcal{L}_k$ is constructed to preserve similarities between the new streaming images and old sampled images, which is also formulated as an inner product loss:

$$\mathcal{L}_k = \left\| \tanh(F(\vec{\boldsymbol{X}}^t; \theta^t))(\tilde{\boldsymbol{B}}^{t*^{\mathrm{T}}}) - k\tilde{\boldsymbol{S}}^t \right\|_F^2, \tag{25}$$

where $\tilde{\boldsymbol{S}}^t$ is computed between the new streaming images and the sampled old images in stage t by the way in (12).

B. Feature Projection Loss

In order to fully utilize the label information of the new streaming images, we construct the feature projection loss

$\mathcal{L}_p$ to project the image features to their corresponding labels, which is formulated as

$$\mathcal{L}_p = \|\vec{\boldsymbol{X}}^{t*}\boldsymbol{W}_p - \vec{\boldsymbol{L}}^t\|_{\mathrm{F}}^2, \tag{26}$$

where $\vec{\boldsymbol{X}}^{t*}$ denotes the image features defined before and $\boldsymbol{W}_p \in R^{d\times c}$ denotes the feature projection matrix.

C. Feature Classification Loss

In addition, we construct the feature classification loss $\mathcal{L}_f$ which is formulated as the cross-entropy loss between the feature probabilities and the labels:

$$\begin{aligned}\mathcal{L}_f &= -\frac{1}{n^t c}\sum_{i=1}^{n^t}\sum_{j=1}^{c}\vec{L}_{i,j}^{\,t}\log P_{i,j}^t \\ &= -\frac{1}{n^t c}\sum_{i=1}^{n^t}\sum_{j=1}^{c}\vec{L}_{i,j}^{\,t}\log\frac{e^{F_{i,j}(\vec{\boldsymbol{x}}_i^t;\varphi^t)}}{\sum_{j=1}^{c}e^{F_{i,j}(\vec{\boldsymbol{x}}_i^t;\varphi^t)}},\end{aligned} \tag{27}$$

where φ^t denotes the image feature learning network parameters in stage t.

We integrate (25), (26) and (27) to get the additional objective function $\mathcal{L}_a$:

$$\mathcal{L}_a = \beta\mathcal{L}_k + \gamma\mathcal{L}_p + \delta\mathcal{L}_f, \tag{28}$$

where β, γ and δ are the hyperparameters for balance.

Overall Objective Function Finally, we combine (20) and (28) to get the overall objective function:

$$\begin{aligned}&\min_{\vec{\theta}^t,\vec{\varphi}^t,\vec{\xi}^t,\vec{\boldsymbol{B}}^t}\ \mathcal{L}_e + \mathcal{L}_a \\ &s.t.\ \vec{\mathbf{B}}^t \in \{-1,+1\}^{n^t\times k}.\end{aligned} \tag{29}$$

Optimization We elaborate the optimization details during the whole training process. In general, it can be described in two phases: label-wise weighs learning and parameters updating. Here, we describe the two phases when the t-th batch of streaming image examples arrive. In the former phase, we first embed the labels by word2vector (Mikolov et al. 2013) to get $\vec{\boldsymbol{Y}}^t$. We then construct a label graph $\mathcal{G}_\mathcal{L}$ by (6) and (7). By feeding $\mathcal{G}_\mathcal{L}$ into the two-layer inductive GNN (Hamilton, Ying, and Leskovec 2017), we obtain the label features $\vec{\boldsymbol{Y}}^{t*}$ and label-wise weights $\boldsymbol{\omega}^t$ by (8). Hence, the affinity matrices $\vec{\boldsymbol{S}}^t$ and $\tilde{\boldsymbol{S}}^t$ are calculated by (12). In the latter phase, we first encode the new streaming images by vision GNN (Han et al. 2022) to get $\vec{\boldsymbol{X}}^t$. We then construct a feature graph $\mathcal{G}_\mathcal{F}$ by (3) and (4) according to the affinity matrix $\vec{\boldsymbol{S}}^t$ learnt in the former phase. By feeding $\mathcal{G}_\mathcal{L}$ into the two-layer inductive GNN, we obtain the image features $\vec{\boldsymbol{X}}^{t*}$ and binary codes $\vec{\boldsymbol{B}}^{t*}$. The hashing network parameters θ^t are updated by minimizing (15) and (25). The image feature learning network parameters φ^t are updated by minimizing (26) and (27). The label feature learning network parameters ξ^t are updated by minimizing (18). In the end, the hash codes of the new streaming images are learnt by minimizing (19) and added to the previous hash codes pool.

Out-of-sample Extension

In the query process, we need to encode the query image to binary code efficiently. We first encode the query image with vision GNN (Han et al. 2022) to get the query feature $\boldsymbol{q}$ in the same way as the training images. Since the label is not accompanied by the query, we construct the query graph $\mathcal{G}_q =< \mathcal{V}_q, \mathcal{E}_q >$ as

$$\mathcal{V}_q = \{\boldsymbol{v}_i \mid \boldsymbol{v}_i = q\}, \tag{30}$$

$$\mathcal{E}_\mathcal{L} = \left\{(\boldsymbol{v}_i,\boldsymbol{v}_j) \mid j \in \hat{\mathrm{sort}}(\cos(\boldsymbol{v}_i,\boldsymbol{v}_j),n)\right\}, \tag{31}$$

where $\hat{\mathrm{sort}}(x,n)$ denotes an descending sorting function to return a set of n indices corresponding to the largest values in x and the cosine similarity $\cos(\boldsymbol{v}_i,\boldsymbol{v}_j)$ between $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ is defined as

$$\cos(\boldsymbol{v}_i,\boldsymbol{v}_j) = \frac{\boldsymbol{v}_i^{\mathrm{T}}\boldsymbol{v}_j}{\|\boldsymbol{v}_i\|\|\boldsymbol{v}_j\|}, \tag{32}$$

where $\|\cdot\|$ denotes the ℓ_2-norm of a vector. By inputting the query graph $\mathcal{G}_q =< \mathcal{V}_q, \mathcal{E}_q >$ into the hash learning network, we can obtain the query's hash code $\boldsymbol{B_q} \in R^k$ efficiently by the following formula:

$$\boldsymbol{B_q} = \mathrm{sgn}(F(\boldsymbol{q};\theta^t)). \tag{33}$$

Experiments

Datasets and Experimental Settings

MIRFlickr (Huiskes and Lew 2008) consists of 25,000 image instances annotated with 24 semantic labels. We remove the rare tag instances which appear less than 20 times, leaving 20,015 instances. A random subset of 2,015 instances are selected as the test and the remaining 18,000 instances are used as both the training and the database. **MSCOCO** (Lin et al. 2014) contains 82,783 training images and 40,504 verification images which belong to 80 categories. After discarding the images without any category information, we use the 122,218 remaining images, in which we randomly select 2,000 images as the test and the remaining 120,218 images are used as the database. A random subset of 40,000 examples from the database is used for training. As for online hash learning, all the images are resized to $224\times224\times3$ and the batch size is set to 50 in the two datasets, because inductive GNNs require constructing the feature graph with an appropriate size.

In the experiments, we compare DGOH with eight state-of-the-art supervised online hashing methods, namely OKH (Huang, Yang, and Zheng 2013), AdaptHash (Cakir and Sclaroff 2015), OSupH (Cakir, Bargal, and Sclaroff 2017), MIHash (Cakir et al. 2017), BSODH (Lin et al. 2019), HMOH (Lin et al. 2020b), SDOH (Zhang et al. 2023) and FOH (Jia et al. 2023). Vision GNN is used to extract the features for all the baselines. We provide the exact values of the parameter configurations in our DGOH as $\alpha = \beta = \eta_l = 0.01$, $\kappa = 1$, $\delta = 0.1$, $\mu = \eta_f = 0.001$, $\lambda = 0.2$, $n = 200$, $m^{t*} = 100$, where η_f and η_l denote the learning rates in the feature network and label network, respectively. The proposed DGOH is trained with Pytorch on a workstation which consists of a CPU with 12 cores, 20 processors

Dataset	Method	Ref.	mAP					Precision@500				
			16	32	48	64	128	16	32	48	64	128
MIRFlickr	OKH	IJCAI'13	0.631	0.631	0.647	0.656	0.667	0.708	0.725	0.744	0.758	0.778
	AdaptHash	ICCV'15	0.579	0.588	0.603	0.605	0.615	0.607	0.635	0.669	0.677	0.707
	OSupH	CVIU'17	0.715	0.742	0.742	0.749	0.744	0.819	0.845	0.847	0.860	0.854
	MIHash	ICCV'17	0.642	0.650	0.636	0.679	0.688	0.656	0.664	0.653	0.726	0.749
	BSODH	AAAI'19	0.589	0.607	0.615	0.615	0.620	0.596	0.643	0.685	0.675	0.687
	HMOH	IJCV'20	0.667	0.683	0.681	0.694	0.697	0.729	0.791	0.791	0.799	0.803
	SDOH	MM'23	0.700	0.709	0.712	0.714	0.717	0.771	0.705	0.732	0.742	0.774
	FOH	AAAI'23	0.741	0.768	0.764	0.775	0.779	0.842	0.861	0.852	0.870	0.866
	DGOH	Ours	**0.852**	**0.875**	**0.880**	**0.885**	**0.887**	**0.942**	**0.946**	**0.951**	**0.947**	**0.946**
MS-COCO	OKH	IJCAI'13	0.413	0.472	0.492	0.498	0.514	0.476	0.482	0.502	0.516	0.531
	AdaptHash	ICCV'15	0.374	0.418	0.430	0.434	0.460	0.418	0.436	0.482	0.486	0.507
	OSupH	CVIU'17	0.477	0.511	0.526	0.543	0.544	0.532	0.548	0.562	0.577	0.579
	MIHash	ICCV'17	0.408	0.403	0.409	0.432	0.433	0.426	0.441	0.457	0.443	0.457
	BSODH	AAAI'19	0.446	0.448	0.467	0.445	0.442	0.458	0.453	0.472	0.483	0.481
	HMOH	IJCV'20	0.472	0.511	0.527	0.531	0.575	0.516	0.539	0.547	0.582	0.596
	SDOH	MM'23	0.470	0.496	0.505	0.508	0.518	0.476	0.519	0.518	0.514	0.569
	FOH	AAAI'23	0.516	0.531	0.531	0.533	0.543	0.580	0.624	0.631	0.636	0.655
	DGOH	Ours	**0.626**	**0.659**	**0.664**	**0.665**	**0.669**	**0.820**	**0.865**	**0.876**	**0.883**	**0.879**

Table 1: The mAP and Precision@500 results of different online hashing methods with various hash bits on two datasets.

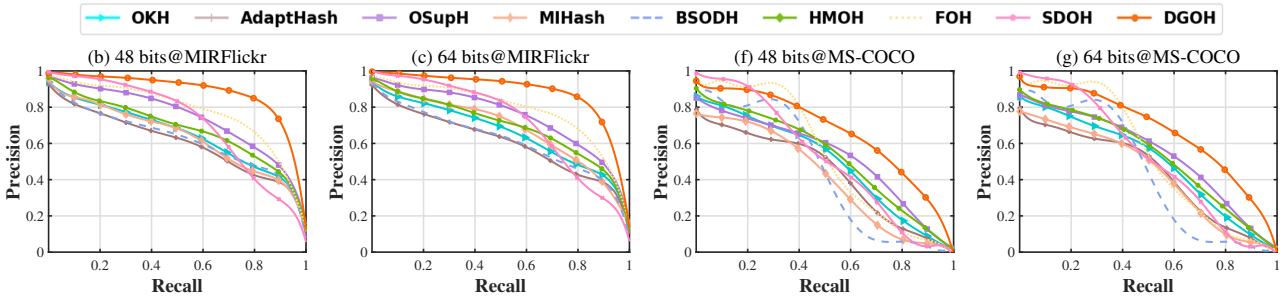

Figure 2: The precision-recall curves with 48 and 64 hash bits on MIRFlickr (b-c) and MS-COCO (f-g).

and an NVIDIA Geforce RTX 3090. We utilize two common accuracy evaluation metrics, i.e., mean average precision (mAP) as well as the mean precision of the top 500 retrieved examples (Precision@500) to verify the effectiveness and evaluate the total training time dealing with all the streaming data to verify the training efficiency of all the methods. The experimental results are the average results of 10 query samplings.

Results and Analyses

Accuracy analysis. We compute mAP and Precision@500 of DGOH and eight state-of-the-art baselines with code lengths varying from 16 to 128 bits on two datasets and summarize the result values in Table **??**. We find that DGOH consistently outperforms all the baselines and achieves up to 11.6% and 13.3% mAP gains on MIRFlickr and MS-COCO, respectively. Figure 2 shows the precision-recall curves of different online hashing methods with 32 to 128 hash bits on two datasets. One can see that DGOH generally outperforms all the baselines by a large margin with different numbers of retrieved points. These results demonstrate the high hash learning power of DGOH holistically.

Time cost analysis. In order to evaluate the efficiency of the proposed DGOH, we record the training time and query time of different online hashing methods with 32 hash bits in Table 2. We observe that the query time is at an average level, but the training time is a little bit higher in comparison with the baselines. This is because DGOH is based on deep networks with a large number of parameters rather than shallow networks with a few parameters. However, the training time is still competitive and feasible to realize online hash learning, because it is unnecessary to update the online hash functions continually once new data arrives in practice. We only need to update them after an amount of data accumulation. In this period, we can just utilize the hash functions produced in the last round, which are also effective.

We analyse that two main factors impact on the training efficiency: label network training and label-wise weights based affinity matrix calculation. Therefore, we conduct experiments on three variants of DGOH: DGOH-l-ω, DGOH+l-ω, DGOH-l+ω^0, where ±l denotes adding the label network or not, -ω denotes constructing the affinity matrix without utilizing the label-wise weights. In this case, we construct the affinity matrix $\hat{\boldsymbol{S}}$ by setting $\hat{\boldsymbol{S}}_{i,j} = 1$ if

Method	MIRFlickr			MS-COCO		
	training	query	mAP	training	query	mAP
OKH	6.87	5.16	0.631	27.0	65.0	0.472
AdaptHash	8.87	6.72	0.588	62.1	78.3	0.418
OSupH	294	6.83	0.742	628	81.1	0.511
MIHash	167	6.84	0.650	337	69.6	0.403
BSODH	32.2	9.74	0.607	84.9	84.8	0.448
HMOH	15.4	9.66	0.683	34.9	97.5	0.511
SDOH	22.3	6.56	0.709	63.1	60.5	0.496
FOH	50.7	3.13	0.768	82.1	29.3	0.531
DGOH-l-$\boldsymbol{\omega}$	37.4	8.52	0.864	87.6	81.8	0.600
DGOH+l-$\boldsymbol{\omega}$	47.3	8.54	0.871	98.4	81.2	0.621
DGOH-l+$\boldsymbol{\omega}^0$	380	8.51	0.871	863	81.6	0.640
DGOH	391	8.51	0.875	877	81.5	0.659

Table 2: The training time (s) and query time per query (ms) comparisons with 32 hash bits on two datasets.

$\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ share at least one common label, otherwise, setting $\hat{\boldsymbol{S}}_{i,j} = 0$. +$\boldsymbol{\omega}^0$ denotes using the initialized label-wise weights to construct the affinity matrix.

From Table 2, we can conclude that the above two factors are indeed time consuming, but why do we still introduce a label network? Reason that we can obtain dramatic accuracy gains by taking them into account. We analyze that existing methods for multi-label image datasets overlook the relationships among the labels, resulting in loss of the label graph structure information to construct the affinity matrix. Besides, the conventional approach of connecting images with common labels leads to excessive parameters, hindering essential image feature learning. Our label network produces label-wise weights to help to construct the affinity matrix, providing more representative image features. In general, in applications where accuracy is not a top priority, you can opt for DGOH-l-$\boldsymbol{\omega}$ which still achieves higher accuracy than all the baselines and shows competitive training time in seconds (the average training time of all the baselines on MIRFlickr is 74.7s).

Parameters Sensitivity Analysis

The hyperparameters may impact on the performance of the proposed DGOH. Hence, we conduct experiments on the hyperparameters, i.e., $\alpha, \beta, \gamma, \delta$ by using different values tuned in $\{1e^{-4}, 1e^{-3}, 1e^{-2}, 1e^{-1}, 1, 10\}$. The results with 32 hash bits on MIRFlickr are shown in Figure 3. Empirically, the optimal hyperparameters are highly robust across different datasets. Hence, they are relatively easy to be adjusted in real applications.

Ablation Study

We implement comprehensive experiments to analyse the contribution of each component of our approach from three aspects. First, we analyse the contributions of the losses. Since $\mathcal{L}_s$, $\mathcal{L}_c$ and $\mathcal{L}_q$ are essential, we show the mAP results on different variants in consideration of the remaining three losses $\mathcal{L}_k$, $\mathcal{L}_f$ and $\mathcal{L}_p$ in Table 3. It is obvious that each loss contributes to the accuracy of DGOH. Second, we analyse the impact of the way to construct the feature graph

$\mathcal{L}_k$	$\mathcal{L}_f$	$\mathcal{L}_p$	MIRFlickr					MS-COCO				
			16	32	48	64	128	16	32	48	64	128
×	×	×	0.844	0.859	0.863	0.869	0.868	0.598	0.618	0.619	0.619	0.628
×	×		**0.855**	0.865	0.871	0.870	0.870	0.599	0.620	0.625	0.623	0.631
×		×	0.848	0.869	0.873	0.873	0.876	0.603	0.638	0.648	0.656	0.649
	×	×	0.846	0.868	0.874	0.875	0.884	0.614	0.638	0.643	0.650	0.654
×			0.851	0.866	0.868	0.879	0.878	0.614	0.644	0.653	0.657	0.650
		×	0.848	0.870	0.876	0.882	0.880	0.614	0.645	0.659	0.663	0.661
	×		0.852	0.869	0.877	0.880	0.874	0.622	0.644	0.649	0.644	0.649
DGOH			0.852	**0.875**	**0.880**	**0.885**	**0.887**	**0.626**	**0.659**	**0.664**	**0.665**	**0.669**
DGOH+$\hat{\boldsymbol{S}}$			0.851	0.866	0.873	0.875	0.868	0.599	0.627	0.635	0.641	0.630
DGOH+NN			0.805	0.826	0.852	0.852	0.832	0.613	**0.673**	0.660	**0.672**	0.638
DOGH-λ			0.837	0.869	0.873	0.878	0.886	0.606	0.634	0.640	0.648	0.651
DGOH-rand			0.852	0.874	0.860	0.867	0.859	0.606	0.637	0.650	0.660	0.664
DGOH-sim			0.835	0.861	0.866	0.864	0.870	0.595	0.633	0.633	0.627	0.634
DGOH-both			0.841	0.856	0.869	0.877	0.878	0.607	0.644	0.659	0.660	0.651
DGOH			**0.852**	**0.875**	**0.880**	**0.885**	**0.887**	**0.626**	0.659	**0.664**	0.665	**0.669**

Table 3: The ablation comparisons on two datasets.

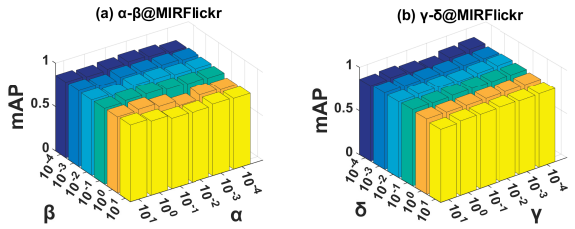

Figure 3: The effects of the hyperparameters with 32 hash bits on MIRFlickr.

$\mathcal{G}_{\mathcal{F}}$. DGOH+$\hat{\boldsymbol{S}}$ denotes constructing $\mathcal{G}_{\mathcal{F}}$ with $\hat{\boldsymbol{S}}$ defined before. DGOH+NN denotes constructing $\mathcal{G}_{\mathcal{F}}$ with a fixed number of neighbors. DGOH-λ denotes constructing $\mathcal{G}_{\mathcal{F}}$ with $\boldsymbol{S}$ without thresholding by λ. As shown in Table 3, DGOH performs the best in most cases. Third, we analyse different strategies to select old examples. DGOH-rand denotes selecting old examples randomly. DGOH-sim denotes selecting similar old examples and DGOH-both denotes selecting both similar and dissimilar old examples. The results verify the effectiveness of our strategy used in DGOH. Note that the first row of results in Table 3 (without no additional loss functions) still performs better than all the baselines which verifies the effectiveness of the deep model compared with the shallow models.

Conclusion

In this paper, we propose a novel Deep Graph Online Hashing (DGOH) method, which for the first time introduces GNNs to realize deep online hashing. Furthermore, we learn label-wise weights dynamically to realize more fine-grained similarity estimation. Finally, an integrated objective function is utilized for training. Our framework can be easily adapted to unsupervised situations bt only using the feature network to construct the feature network when switching it to unsupervised scenarios. Experimental results verify the effectiveness and efficiency of the proposed DGOH method.

Acknowledgements

This work was supported in part by the grant of the National Science Foundation of China under Grant Nos. 62202438, 62172090; the Natural Science Foundation of Shandong Province Grant No. ZR2024MF128; Start-up Research Fund of Southeast University under Grant RF1028623097. We thank the Big Data Computing Center of Southeast University for providing the facility support on the numerical calculations.

References

Cakir, F.; Bargal, S. A.; and Sclaroff, S. 2017. Online supervised hashing. In *Proceedings of the Computer Vision and Image Understanding*, volume 156, 162–173.

Cakir, F.; He, K.; Adel Bargal, S.; and Sclaroff, S. 2017. Mihash: Online hashing with mutual information. In *Proceedings of the IEEE International Conference on Computer Vision*, 437–445.

Cakir, F.; and Sclaroff, S. 2015. Adaptive hashing for fast similarity search. In *Proceedings of the IEEE International Conference on Computer Vision*, 1044–1052.

Chen, X.; King, I.; and Lyu, M. R. 2017. FROSH: FasteR Online Sketching Hashing. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 1–10.

Datar, M.; Immorlica, N.; Indyk, P.; and Mirrokni, V. S. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the Annual Symposium on Computational Geometry*, 253–262.

Fang, Y.; Zhang, H.; and Liu, L. 2021. Label projection online hashing for balanced similarity. *Journal of Visual Communication and Image Representation*, 80: 103314.

Gui, J.; Liu, T.; Sun, Z.; Tao, D.; and Tan, T. 2017. Fast supervised discrete hashing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(2): 490–496.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Proceedings of the Advances in Neural Information Processing Systems*, volume 30.

Han, K.; Wang, Y.; Guo, J.; Tang, Y.; and Wu, E. 2022. Vision gnn: An image is worth graph of nodes. *arXiv preprint arXiv:2206.00272*.

Huang, L.-K.; Yang, Q.; and Zheng, W.-S. 2013. Online hashing. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1422–1428.

Huiskes, M. J.; and Lew, M. S. 2008. The mir flickr retrieval evaluation. In *Proceedings of the ACM International Conference on Multimedia Information Retrieval*, 39–43.

Jia, W.; Cao, Y.; Liu, J.; and Gui, J. 2023. Fast Online Hashing with Multi-Label Projection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37.

Jin, S.; Zhou, Q.; Yao, H.; Liu, Y.; and Hua, X.-S. 2021. Asynchronous teacher guided bit-wise hard mining for online hashing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 1717–1724.

Leng, C.; Wu, J.; Cheng, J.; Bai, X.; and Lu, H. 2015. Online sketching hashing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2503–2511.

Lin, M.; Ji, R.; Chen, S.; Sun, X.; and Lin, C.-W. 2020a. Similarity-preserving linkage hashing for online image retrieval. *IEEE Transactions on Image Processing*, 29: 5289–5300.

Lin, M.; Ji, R.; Liu, H.; Sun, X.; Chen, S.; and Tian, Q. 2020b. Hadamard matrix guided online hashing. *International Journal of Computer Vision*, 128: 2279–2306.

Lin, M.; Ji, R.; Liu, H.; Sun, X.; Wu, Y.; and Wu, Y. 2019. Towards optimal discrete online hashing with balanced similarity. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 8722–8729.

Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft coco: Common objects in context. In *Proceedings of the European Conference on Computer Vision*, 740–755. Springer.

Liu, X.; Yi, J.; Cheung, Y.-m.; Xu, X.; and Cui, Z. 2022. Omgh: Online manifold-guided hashing for flexible cross-modal retrieval. *IEEE Transactions on Multimedia*.

Lu, X.; Zhu, L.; Cheng, Z.; Nie, L.; and Zhang, H. 2019. Online multi-modal hashing with dynamic query-adaption. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, 715–724.

Luo, X.; Wang, H.; Wu, D.; Chen, C.; Deng, M.; Huang, J.; and Hua, X.-S. 2023. A survey on deep hashing methods. *ACM Transactions on Knowledge Discovery from Data*, 17(1): 1–50.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

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

Wang, Y.; Song, J.; Zhou, K.; and Liu, Y. 2021. Unsupervised deep hashing with node representation for image retrieval. *Pattern Recognition*, 112: 107785.

Weng, Z.; and Zhu, Y. 2020. Online hashing with efficient updating of binary codes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 12354–12361.

Wu, X.-M.; Luo, X.; Zhan, Y.-W.; Ding, C.-L.; Chen, Z.-D.; and Xu, X.-S. 2022. Online Enhanced Semantic Hashing: Towards Effective and Efficient Retrieval for Streaming Multi-Modal Data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 4263–4271.

Xia, R.; Pan, Y.; Lai, H.; Liu, C.; and Yan, S. 2014. Supervised hashing for image retrieval via image representation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28.

Xie, L.; Shen, J.; and Zhu, L. 2016. Online cross-modal hashing for web image retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Xie, Y.; Zeng, X.; Wang, T.; and Yi, Y. 2022. Online deep hashing for both uni-modal and cross-modal retrieval. *Information Sciences*, 608: 1480–1502.

Zhan, Y.-W.; Luo, X.; Sun, Y.; Wang, Y.; Chen, Z.-D.; and Xu, X.-S. 2021. Weakly-supervised online hashing. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, 1–6. IEEE.

Zhang, C.-Y.; Luo, X.; Zhan, Y.-W.; Zhang, P.-F.; Chen, Z.-D.; Wang, Y.; Yang, X.; and Xu, X.-S. 2023. Self-Distillation Dual-Memory Online Hashing with Hash Centers for Streaming Data Retrieval. In *Proceedings of the ACM International Conference on Multimedia*, 6340–6349.

Zhang, D.; Wu, X.-J.; and Chen, G. 2023. ONION: Online Semantic Autoencoder Hashing for Cross-Modal Retrieval. *ACM Transactions on Intelligent Systems and Technology*, 14(2): 1–18.