Improved Deep Unsupervised Hashing via Prototypical Learning

Zeyu Ma*
Harbin Institute of Technology
Shenzhen, China
zeyu.ma@stu.hit.edu.cn

Wei Ju*
Peking University
Beijing, China
juwei@pku.edu.cn

Xiao Luo†
Peking University
Beijing, China
xiaoluo@pku.edu.cn

Chong Chen
Peking University
Beijing, China
cheung1990@126.com

Xian-Sheng Hua
Zhejiang University
Hangzhou, China
huaxiansheng@gmail.com

Guangming Lu†
Guangdong Provincial Key
Laboratory of Novel Security
Intelligence Technologies
Harbin Institute of Technology
Shenzhen, China
luguangm@hit.edu.cn

ABSTRACT

Hashing has become increasingly popular in approximate nearest neighbor search in recent years due to its storage and computational efficiency. While deep unsupervised hashing has shown encouraging performance recently, its efficacy in the more realistic unsupervised situation is far from satisfactory due to two limitations. On one hand, they usually neglect the underlying global semantic structure in the deep feature space. On the other hand, they also ignore reconstructing the global structure in the hash code space. In this research, we develop a simple yet effective approach named dee**P** Unsupe**R**vised hashing via **P**rototypical **LE**arning (PURPLE). Specifically, PURPLE introduces both feature prototypes and hashing prototypes to model the underlying semantic structures of the images in both deep feature space and hash code space. Then we impose a smoothness constraint to regularize the consistency of the global structures in two spaces through our semantic prototypical consistency learning. Moreover, our method encourages the prototypical consistency for different augmentations of each image via contrastive prototypical consistency learning. Comprehensive experiments on three benchmark datasets demonstrate that our proposed PURPLE performs better than a variety of state-of-the-art retrieval methods.

CCS CONCEPTS

• **Information Systems** → **Information Search; Similarity measures**; • **Theory of computation** → **Unsupervised learning and clustering**.

*Equal contribution.
†Corresponding authors.

MM '22, October 10–14, 2022, Lisboa, Portugal

ACM ISBN 978-1-4503-9203-7/22/10…$15.00
https://doi.org/10.1145/3503161.3548403

KEYWORDS

Deep hashing, image retrieval, unsupervised learning

ACM Reference Format:
Zeyu Ma, Wei Ju, Xiao Luo, Chong Chen, Xian-Sheng Hua, and Guangming Lu. 2022. Improved Deep Unsupervised Hashing via Prototypical Learning. In *Proceedings of the 30th ACM International Conference on Multimedia (MM '22), October 10–14, 2022, Lisboa, Portugal.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3503161.3548403

1 INTRODUCTION

Large-scale image search has been a important problem in the multimedia community. Among various search algorithms, learning to hash has achieved increasing interest thanks to its excellent efficiency [31, 35]. The principle of hashing algorithms is compressing high-dimensional input samples into compact hash codes, which maintains the similarity information of the original data points [31]. As the popularity of deep learning, a range of researchers combine supervised hashing and deep neural networks, which achieves remarkable performance in image retrieval by generating well similarity preserved hash codes with the help of semantic labels [2, 3, 21, 23, 29, 35, 42, 45, 50].

Despite the great success, supervised hashing approaches are hard to be deployed in practice owing to the expense of large-scale data annotations. Accordingly, plenty of unsupervised methods have been developed and provide a cost-effective solution to this problem [28, 36, 40, 46]. A two-step framework is employed by many recent unsupervised hashing methods: (1) The local semantic similarity structure can be built based on the extracted deep features through the pre-trained neural network. (2) A hashing network can be optimized under the supervision of the acquired similarity structure to generate compact hash codes for efficient image retrieval.

Nevertheless, existing methods [28, 36, 40, 46] suffer from the following limitations that could influence the quality of hash codes. First, they mostly focus on constructing the local semantic similarity relationships but fail to discover the underlying global semantic structure over the whole data distribution. It is worth noticing that images in a dataset should possess a global semantic structure. For the entire dataset, deep features usually accumulate around

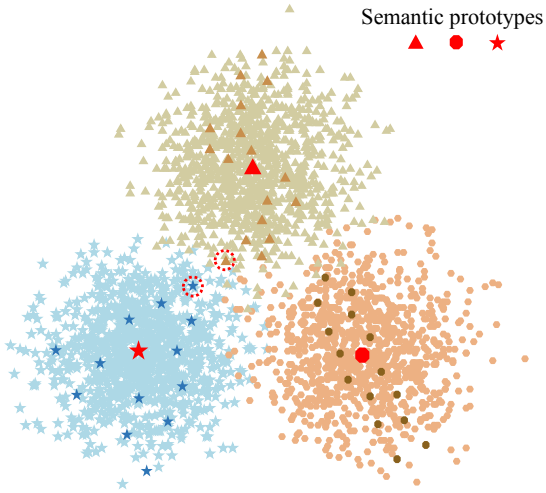

Figure 1: An illustration of the global semantic structure in the deep feature space. Different colors indicate different semantics.

their feature prototypes corresponding to their semantics. However, existing methods [28, 36, 40, 46] mostly do not explore semantic structure from the global view. As shown in Figure 1, the features of input images containing similar semantics should be close to their corresponding prototypes. The distance of the blue data point and the brown data point circled in the dashed line is small although they contain dissimilar semantics based on the global semantic structure. The similarity relationships of image pairs are likely to be misjudged if ignoring the global semantic structure. Second, effective hash codes should also reflect the global semantic structure of the dataset in the hash code space, so that the images with similar semantic properties (i.e., the same class label) can be compactly embedded in the hash code space. Ignoring reconstructing global semantic structure in the hash code space directly affects the efficiency of the retrieval performance.

Accordingly, we propose a new deep unsupervised hashing method named dee**P** **U**nsupe**R**vised hashing via **P**rototypical **LE**arning (PURPLE), which introduces prototypes in both the deep feature space and the hash code space to explore and preserve the global semantic structure respectively, and obtain robust hash codes by two kinds of consistency learning. Specifically, PURPLE firstly constructs the global semantic structure by mapping deep features to a set of prototypes and then obtaining the similarity graph based on the pseudo-labels of images. Then a novel semantic prototypical consistency loss is adopted to match the hashing graph with the similarity graph obtained in the deep feature space. Moreover, we build the hashing prototypes (i.e., cluster centroids in the hash code space) based on the Hadamard matrix or random sampling, which depict the hashing semantic structure of the entire dataset. Then the Sinkhorn-knopp algorithm is utilized to generate soft hashing prototypical assignments with necessary constraints. In the end, the contrastive prototypical semantic consistency in the hash code space is achieved by enforcing the prototypical assignment consistency between different augmentations of each image.

The contributions of this paper are summarized as:

- We introduce both feature prototypes and hashing prototypes to model the underlying semantic structures of images in both deep feature space and hash code space.
- We propose a semantic prototypical consistency loss and a contrastive prototypical consistency loss to encourage both the semantic structure alignment of two spaces and the prototypical assignment consistency of different views of each image in the hash code space.
- Extensive experiments on three popular benchmark datasets demonstrate that our PURPLE outperforms recent state-of-the-art unsupervised hashing methods.

2 RELATED WORK

2.1 Deep Unsupervised Hashing

Many recent deep unsupervised hashing methods solve the unsupervised problem by generating the semantic structure of training samples based on their deep features [33, 34, 49]. To be specific, DeepBit [28] tries to learn hash codes by preserving the similarities between the similar pairs of images and their corresponding rotated images. SSDH [46] constructs the similarity structure based on Gaussian estimation and further preserves the semantic structure in the hamming space. DistillHash [47] utilizes a Bayes optimal classifier to help distill the image pairs with confident similarity signals and thus enhances the generated semantic structure. CUDH [13] generates hash codes based on the aggregated clusters, which are recursively learned by the soft clustering model. MLS^{3}RDUH [40] reconstructs the local semantic similarity structure based on a novel similarity matrix by using the manifold structure in feature space and the similarity of datapoints. SPQ [19] generates hash codes by utilizing the cross quantized contrastive learning with data augmentations from the view of self-supervised learning. Our model further improves the performance of deep unsupervised hashing by exploring the global semantic structure in both the deep feature space and the hash code space.

2.2 Self-supervised Contrastive Learning

Recent works [7, 11, 16, 22, 24, 26, 27, 44] show that unsupervised image representation learning has achieved significant improvement based on contrastive learning. [14] tries to contrast positive pairs with negative pairs for representation learning. SimCLR [7] conducts contrastive learning on elements in the same batch without requiring a memory bank, which improves the performance on ImageNet. Recent works [19, 34] have integrated contrastive learning with deep unsupervised hashing, taking into account that hash code is a type of representation. However, most downstream tasks suffer from the burden of computation and storage when comparing pairs of image feature representations. Benefiting from the low storage cost of hashing methods, contrastive learning can be well employed in our model to help enhance the performance of unsupervised hashing. Inspired by recent methods in contrastive learning, we develop the semantic prototypical consistency learning and the contrastive prototypical consistency learning to help improve the performance of image retrieval.

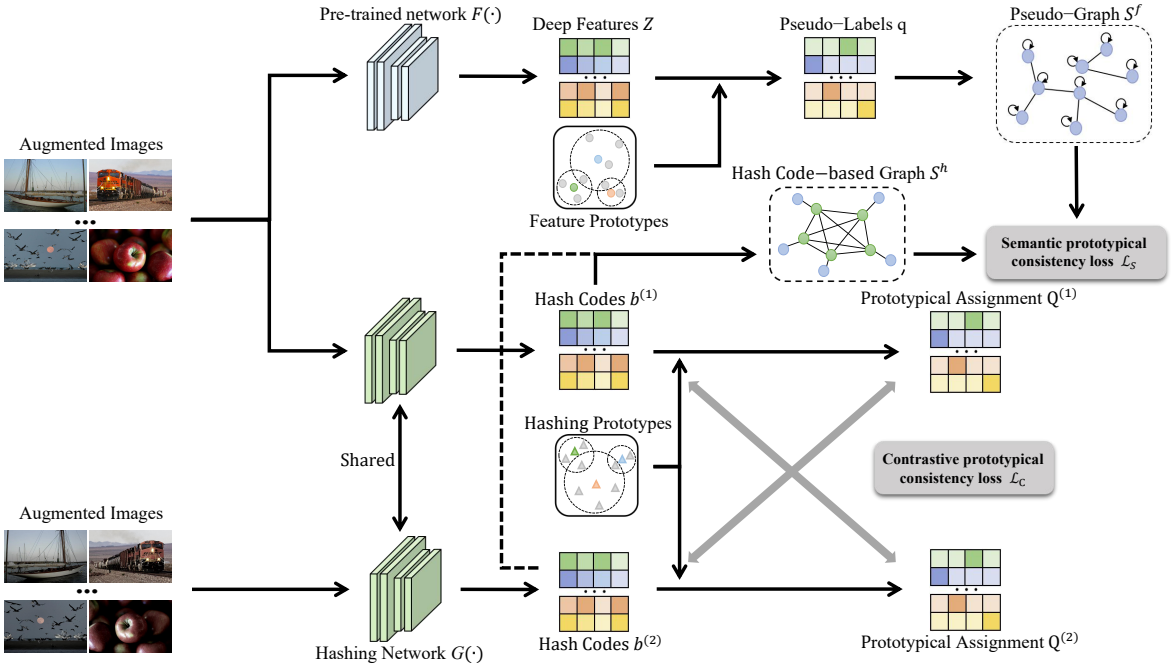

Figure 2: Illustration of our proposed framework PURPLE. On the one hand, PURPLE generates the semantic structure information by constructing the feature-based pseudo-graph S^f, which guides the hashing-based semantic structure-preserving together with the hash code-based graph S^h through the semantic prototypical consistency learning. On the other hand, contrastive consistency learning is based on the prototypical assignment results of different augmentations of each image.

2.3 Deep Clustering

Our method is also related with deep clustering [1, 5, 17, 25, 51] since the global semantic structure can be reflected by clustering to some extent. Among them, DeepCluster [5] utilizes pseudo-labels generated based on k-means clustering assignments as the visual representations. [1] further utilizes the optimal transport theory to optimize the pseudo-label assignments. PICA [17] maximizes the partition confidence of the cluster solution to learn the most semantically plausible data separation. Inspired by contrastive learning, a range of works enhance deep clustering by regarding the pseudo-label as a degraded representation, which can achieve promising results by mapping input samples into the subspace with the same dimensionality as the cluster number [25, 51]. There are also some works in different fields [15, 30] attempting to employ deep clustering to enhance representation learning by considering the global semantic structure in the deep feature space. Among them, SwAV [6] contrasts cluster assignments from different views for self-supervised representation learning. Different from them, our method utilizes hashing prototypes to explore the global semantic structure in the hash code space.

3 METHODOLOGY

In this section, we introduce the problem definition of unsupervised hashing and then illustrate our method PURPLE in Figure 2, including its network architecture, semantic prototypical consistency learning, and contrastive prototypical consistency learning.

3.1 Problem Definition

For unsupervised hashing learning, $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$ represents the training set with N unlabeled images. A hash function is aimed to be learned $\mathcal{H} : \mathbf{x} \rightarrow \mathbf{b} \in \{-1, 1\}^L$, where $\mathbf{x}$ represents the input sample and $\mathbf{b}$ represents the learned compact L-bit hash code. This process is expected to maintain similarity of datapoints. Namely, samples with similar semantic labels are expected to have corresponding hash codes with small Hamming distance.

3.2 Network Architecture

As shown in Figure 2, our model contains two modules for deep feature extraction and hash code learning, respectively. To be specific, following [19, 40, 46], we utilize a pre-trained network (e.g., VGG-F [39]) removing the last layer $F(\cdot)$ to extract deep features of training images, and reconstruct the semantic structure in the feature space. Besides, the hashing network $G(\cdot)$ is converted by replacing the last layer of the pre-trained network with a fully-connected layer, which contains L units for hash code learning.

3.3 Semantic Prototypical Consistency Learning

It is worth noticing that images in a dataset should possess a global semantic structure. In our model, global semantic structures are modeled in both the deep feature space and the hash code space. We first generate the semantic structure in the pre-trained deep

feature space, and then attempt to preserve the semantic structure information in the hash code learning process.

3.3.1 Feature-based Semantic Structure Generating. To comprehensively explore the semantic structure of the training set $\mathcal{X}$, given the pre-trained deep features[1] $Z = \{\mathbf{z}_i = F(\mathbf{x}_i)\}_{i=1}^{N}$ (after L_2 normalization), we assume that the deep features of images with the same semantics are expected to accumulate in the latent space. Specifically, we introduce a set of M prototypes $C = \{\mathbf{c}_1, ..., \mathbf{c}_M\}$ to represent the feature centroids in the latent space. In practice, we use the k-means clustering to get the prototype set. Then we match all the features to the M prototypes, the similarity between each deep feature $\mathbf{z}_i$ and the m-th prototype $\mathbf{c}_m$ can be formalized as:

$$(\mathbf{q}_i)_m = \frac{\exp(\mathbf{z}_i^\top \mathbf{c}_m/\tau)}{\sum_{m'}^{M} \exp(\mathbf{z}_i^\top \mathbf{c}_{m'}/\tau)} \tag{1}$$

where $\mathbf{q}_i$ denotes the pseudo-label of the i-th sample and τ is the temperature parameter. Given the pseudo-labels of I unlabeled samples in a mini-batch, we can construct the deep feature-based pseudo-graph $S^f \in \mathbb{R}^{I\times I}$ for the batch in the following formulation:

$$S_{ij}^f = \begin{cases} 1 & \text{if } \mathbf{q}_i^\top \mathbf{q}_j \geq T \\ \mathbf{q}_i^\top \mathbf{q}_j & \text{otherwise} \end{cases} \tag{2}$$

where T is a threshold parameter and sample pairs with similarity larger than T are fully connected in S^f, and each sample is fully connected to itself.

3.3.2 Hashing-based Semantic Structure Preserving. The semantic structure consistency learning aims to learn hash codes guided by the underlying semantics in the deep feature space, such that obtained hash codes can preserve the semantic structure of the training images. To be specific, images with similar semantics are expected to be mapped to similar hash codes.

At the feature-based semantic structure constructing step, the pseudo-graph S^f is generated, which can be used as the guidance of hash code learning through constructing a hash code-based graph S^h. To construct the graph S^h, two augmented views of each image $\mathbf{x}_i$ are utilized and their hash codes can be obtained through the hashing network $G(\cdot)$, which can be denoted as $\{\mathbf{b}_i^{(1)}, \mathbf{b}_i^{(2)}\}_{i=1}^{I}$. Then the hash code similarity graph $S_{ij}^h \in \mathbb{R}^{I\times I}$ can be derived by the following formulation:

$$S_{ij}^h = \begin{cases} e^{\mathbf{b}_i^{(1)} \star \mathbf{b}_i^{(2)}/\tau} & i = j, \\ e^{\mathbf{b}_i^{(1)} \star \mathbf{b}_j^{(1)}/\tau} & i \neq j \end{cases} \tag{3}$$

where the $\star$ denotes the cosine similarity metric, i.e., $\mathbf{a}\star\mathbf{b} = \frac{\mathbf{a}^T\mathbf{b}}{||\mathbf{a}||||\mathbf{b}||}$. For the purpose of preserving the semantic structure information, we aim to train the hashing network with the guidance of the pseudo-graph. Accordingly, the semantic prototypical consistency loss is defined to minimize the weighted cross-entropy between two graphs, which is formulated as:

$$\mathcal{L}_S(S^f, S^h) = \frac{1}{I}\sum_{i,j=1}^{I}\left(-S_{ij}^f \log\left(\frac{S_{ij}^h}{\sum_{j'=1}^{I} S_{ij'}^h}\right)\right) \tag{4}$$

[1]Here we omit data augmentation for brevity.

where each term corresponds to the element-wise calculation in corresponding matrices.

3.4 Contrastive Prototypical Consistency Learning

In the hash code space, we can also expect the global semantic structure. Specifically, we seek to construct M hashing prototypes $\{\mathbf{h}_1, ..., \mathbf{h}_M\}$ well separated in the hash code space, each of which implies a cluster centroid. Inspired by [48], we observe that hashing prototypes learned from unlabeled data with diverse mutual Hamming distances perform worse than hashing prototypes with given Hamming distance. As a result, we leverage the Hadamard matrix to generate prototypes in the hash code space. Note that the code length L can usually be written as $L = 2^K$ following [19, 31, 36], so we can construct the Hadamard matrix $\mathbf{H}_{2^K} \in \{-1, 1\}^{2^K\times 2^K}$ as follows:

$$\begin{aligned} \mathbf{H}_2 &= \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \\ \mathbf{H}_{2^k} &= \begin{bmatrix} \mathbf{H}_{2^{k-1}} & \mathbf{H}_{2^{k-1}} \\ \mathbf{H}_{2^{k-1}} & -\mathbf{H}_{2^{k-1}} \end{bmatrix} \quad (K-1 \geq k \geq 2) \end{aligned} \tag{5}$$

where the column vectors are mutually orthogonal. Then, we utilize the Hadamard matrix to generate desirable hashing prototypes. To be specific, the binary column vectors sampled from $\mathbf{H}_{2^K}$ are used as hashing prototypes. As a result, if the number of hashing prototypes is smaller than the hash code length, we can produce hashing prototypes using the above strategy. The distance between any two prototypes is 2^{K-1}. However, when the number of hashing prototypes M is larger than the hash code length L, the strategy cannot work. In this case, we randomly sample the bits of each hashing prototype instead. To be specific, each bit of a hashing prototype comes from the Bernoulli distribution, i.e., Bern(0.5). It can be shown that the expected distance between these hashing prototypes is 2^{K-1}, i.e., half of the code length [48]. Then, we leverage the produced hashing prototypes to guide the optimization of the hashing network.

Inspired by recent clustering-based self-supervised methods [6], we contrast multiple views of the each image by comparing their prototypical assignments in the hash code space. Specifically, given two different augmentations of the same image $\mathbf{x}_i$, we can compute the probability of the hash code $\mathbf{b}_i^{(r)} = sign(G(\mathbf{x}_i^{(r)}))$ being assigned to the m-th hashing prototype $\mathbf{h}_m$ by comparing hash codes with a set of hashing prototypes as follows:

$$p(y = m|\mathbf{b}_i^{(r)}) = \frac{\exp(\mathbf{b}_i^{(r)^\top}\mathbf{h}_m/\tau)}{\sum_{m'=1}^{M}\exp(\mathbf{b}_i^\top \mathbf{h}_{m'}/\tau)} \tag{6}$$

where $r = 1$ or 2 and y represents the prototypical assignment label to the prototype $\mathbf{h}_m$ and τ is the temperature parameter. To encourage the prototypical consistency between two correlated views (i.e., $\mathbf{x}_i^{(1)}$ and $\mathbf{x}_i^{(2)}$), we expect to predict the hashing prototypical assignments of $\mathbf{b}_i^{(2)}$ ($\mathbf{b}_i^{(1)}$) with the hash code $\mathbf{b}_i^{(1)}$ ($\mathbf{b}_i^{(2)}$) from the correlated view. Formally, we define the contrastive prototypical consistency objective function via minimizing the average cross-entropy loss between the prototypical assignment result and the

probability :

$$\ell(\mathbf{b}_i^{(2)}, \mathbf{b}_i^{(1)}) = -\sum_{m=1}^{M} q(y=m|\mathbf{b}_i^{(1)}) \log p(y=m|\mathbf{b}_i^{(2)}) \quad (7)$$

where $q(y|\mathbf{b}_i^{(1)})$ denotes the target hashing prototypical assignments of $\mathbf{x}_i^{(1)}$. The other loss function can be derived similarly. The final contrastive prototypical consistency learning loss is written as:

$$\mathcal{L}_C = \frac{1}{2I}\sum_{i=1}^{I} \ell(\mathbf{b}_i^{(2)}, \mathbf{b}_i^{(1)}) + \ell(\mathbf{b}_i^{(1)}, \mathbf{b}_i^{(2)}) \quad (8)$$

Contrastive prototypical consistency learning aims to achieve consistent hash codes for different image views by comparing their prototypical assignments instead of their hash codes. Inspired by recent studies [1, 6], we calculate the target hashing prototypical assignment $q(y|\mathbf{b}_i)$, and then conduct the gradient descend algorithm to optimize the hashing network. To begin, we add necessary constraints when calculating the target prototypical assignments to avoid the issue of potential trivial solutions [6]. In the following, we omit the superscript of $q(y|\mathbf{b}_i^{(r)})$. We first seek to ensure that the target hashing prototypical assignment $q(y|\mathbf{b}_i)$ for each image $\mathbf{x}_i$ are an one-hot vector. Moreover, we require the assignment balance for each hashing prototype using $q\left(y=m \mid \mathbf{b}_i\right)=\frac{I}{M}$, which indicates the uniformity in the hash code space.

Note that obtaining $q\left(y=m \mid \mathbf{b}_i\right)$ is equivalent to solving an optimal transport problem [1]. Thus, we leverage an alternative optimization algorithm. To begin, we introduce a matrix $\mathbf{Q} \in \mathbb{R}^{M\times I}$ with $\mathbf{Q}_{mi} = \frac{1}{I} q\left(y=m \mid \mathbf{b}_i\right)$. To achieve the uniformity in the hash code space, we encourage $\mathbf{Q}$ to be a transportation polytope as:

$$\Omega = \left\{\mathbf{Q} \in \mathbb{R}_+^{M\times I} | \mathbf{Q}1_I = \frac{1}{M}1_M, \mathbf{Q}^T 1_M = \frac{1}{I}1_I\right\} \quad (9)$$

where $1_I \in \mathbb{R}^I$ is a all-one vector and the problem is solved within a minibatch for better efficiency. Then, we maximize the overall similarity between each hash code and its corresponding hashing prototype in a batch as follows:

$$O = \sum_{i=1}^{I}\sum_{m=1}^{M} q\left(y=m \mid \mathbf{b}_i\right) h_m^T \mathbf{b}_i = I \cdot tr(\mathbf{B}^T \mathbf{H}\mathbf{Q}) \quad (10)$$

where $\mathbf{H} = [\mathbf{h}_1, \cdots, \mathbf{h}_M] \in \mathbb{R}^{L\times M}$ and $\mathbf{B} = [\mathbf{b}_1, \cdots, \mathbf{b}_I] \in \mathbb{R}^{L\times I}$.

The solution to maximizing Eq. 10 with constraint in Eq. 9 can be derived by the Sinkhorn-Knopp Algorithm [6, 9, 15, 30]. In formulation, after adding a regularization term $-I\gamma \sum_{ij} \mathbf{Q}_{ij} \log \mathbf{Q}_{ij}$ [6], given $\mathbf{P} = \exp(\frac{\mathbf{H}^T\mathbf{B}}{\gamma})$, where γ is a parameter set to 0.05 empirically,

Algorithm 1 Sinkhorn-Knopp Algorithm

Input: The matrix $\mathbf{P} \in \mathbb{R}^{M\times I}$;
Output: The target probability matrix $\mathbf{Q}^*$;

$\hat{\mathbf{Q}} = \mathbf{P}$.
while not convergence **do**
  Row renomalization of $\hat{\mathbf{Q}}$ to ensure $\hat{\mathbf{Q}}1_I = \frac{1}{M}1_M$.
  Column renormlization of $\hat{\mathbf{Q}}$ to ensure $\hat{\mathbf{Q}}^T 1_M = \frac{1}{I}1_I$.
end while

Algorithm 2 Learning Algorithm of PURPLE

Input: Training images: $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$, the hash code length: L, the number of prototypes: M, the batch size: I;
Output: Parameters Θ of the hashing network $G(\cdot)$;

Initialize the hashing network from the pre-trained network.
Generate M hashing prototypes using Hadamard matrix or sampling strategy.
while not convergence **do**
  Sample I images from $\mathcal{X}$ and produce their augmented views to form a mini-batch.
  Construct the deep feature-based pseudo-graph by Eq. 2.
  Calculate the outputs by forward-propagation through the network $G(\cdot)$.
  Get the target assignment result through Algorithm 1.
  Calculate the loss by Eq.12.
  Update parameters of $G(\cdot)$ through back propagation.
end while

the optimal $\hat{\mathbf{Q}}$ can be derived by Algorithm 1. Our preliminary experiments show that employing three iterations can achieve great performance with less computational cost, and that using soft hashing target assignments has a better performance than using one-hot predictions. The potential reason can be that one-hot predictions could introduce much noise while losing part of information. As a result, the soft hashing target assignments in Eq. 7 can be calculated as follows:

$$\hat{q}\left(y=m \mid \mathbf{b}_i\right) = I \cdot \hat{\mathbf{Q}}_{mi}, \quad (11)$$

and then we update the parameter of the hashing network in an iterative manner.

Our method contributes to the hash code learning in the following aspects: (1) The contrastive prototypical consistency learning prompts the consistency of the prototypical assignments of different augmented views for each sample, which makes for generating hash codes robust to the perturbation. (2) Benefit from the uniformity of prototypes in the hash code space, we can generate well-separated hash codes, which maximize the capacity of the hash code space. (3) Because contrasting the cluster assignments of examples has been shown to generate better feature representations for downstream computer vision tasks [6, 37], our model can be expected to generate discriminative binary descriptors for effective large-scale image retrieval. Finally, the ultimate loss of the hashing network learning is formulated as follows:

$$\mathcal{L} = \mathcal{L}_S + \mathcal{L}_C \quad (12)$$

The Algorithm 2 provides a summary of the entire learning procedure of the hashing network. Unfortunately, for non-zero inputs, the derivation of the $sign(\cdot)$ could be zero and it is not differentiable at the zero point, which could accordingly prevent the back-propagation algorithm from updating the parameters of the model when minimizing the Eq. 12. Instead, the $tanh(\cdot)$ is utilized to approximate the result of the $sign(\cdot)$. The approximate hash codes can be generated by $v_i^{(r)} = tanh(G(\mathbf{x}_i^{(r)}))$ to replace $\mathbf{b}_i^{(r)}$ in the above equations. We use the mini-batch standard stochastic gradient descent (SGD) method to minimize the loss objective.

Table 1: MAP results for different methods on CIFAR-10, FLICKR25K and NUS-WIDE.

Methods	CIFAR-10			FLICKR25K			NUS-WIDE		
	16bits	32bits	64bits	16bits	32bits	64bits	16bits	32bits	64bits
LSH [12]	0.132	0.137	0.145	0.583	0.589	0.593	0.432	0.441	0.443
SH [43]	0.161	0.158	0.151	0.591	0.592	0.602	0.510	0.512	0.518
DeepBit [28]	0.220	0.241	0.252	0.593	0.593	0.620	0.454	0.463	0.477
SGH [10]	0.180	0.183	0.189	0.616	0.628	0.625	0.593	0.590	0.607
SSDH [46]	0.257	0.256	0.259	0.627	0.633	0.656	0.580	0.593	0.610
DistillHash [47]	0.284	0.285	0.287	0.696	0.706	0.708	0.667	0.675	0.677
CUDH [13]	0.286	0.290	0.303	0.661	0.675	0.683	0.693	0.709	0.722
MLS^{3}RDUH [40]	0.288	0.296	0.314	0.697	0.701	0.708	0.713	0.727	0.750
TBH [38]	0.293	0.310	0.323	0.702	0.714	0.720	0.717	0.725	0.735
GLC [32]	0.316	0.330	0.305	0.758	0.772	0.777	0.759	0.772	0.783
SPQ [19]	0.308	0.377	0.409	0.757	0.769	0.778	0.766	0.774	0.785
PURPLE (Ours)	**0.515**	**0.520**	**0.544**	**0.814**	**0.828**	**0.831**	**0.799**	**0.802**	**0.808**

4 EXPERIMENT

Extensive experiments are conducted to evaluate PURPLE compared with several state-of-the-art unsupervised hashing methods.

4.1 Datasets and Setup

CIFAR-10 [20] contains 60000 images of ten unique categories. 1000 images are randomly selected from each class as the query set, and the rest images are utilized as the retrieval set, from which 500 images are randomly sampled from each class as the training set.
FLICKR25K [18] consists of 25,000 images labeled by at least one of the 24 categories. 2,000 randomly selected images are used as the query set with the remaining images as the retrieval set, where we randomly select 5000 images as the training set.
NUS-WIDE [8] has 269,648 images of 81 categories. Following [19, 38], we use the subset that contains the 21 most popular categories. 100 images are randomly selected from each class as the query set with the remaining images as the retrieval set, from which We randomly select 500 images for each category as the training set.

We compare our PURPLE with state-of-the-art methods including two traditional shallow methods (i.e., LSH [12] and SH [43]) and nine deep learning-based methods (i.e., SGH [10], DeepBits [28], SSDH [46], DistillHash [47], CUDH [13], MLS^{3}RUDH [40], TBH [38], GLC [32] and SPQ [19]). For our method, we adopt the same hashing network as [19]. For baselines, we follow the settings in their corresponding papers [19, 32, 38, 40].

We implement PURPLE by Pytorch V1.8.0 on an NVIDIA GeForce RTX 3090 GPU. Our model is optimized by mini-batch SGD with momentum and the mini-batch size is set as 48. The learning rates of the backbone and the newly added fully connected layer are fixed at 0.00001 and 0.001, respectively. We resize images to 224×224 as the training inputs. Data augmentation in the experiments contains random cropping and resizing, color jitter, random grayscale, Gaussian blur and random horizontal flip [16]. The temperature parameter τ is set as 0.5. The number of prototypes M and the threshold parameter T are set as 50 and 0.8 as default, respectively.

We construct the ground-truth similarity information based on the image labels for evaluation. To be specific, for CIFAR-10, two images are regarded as similar when they have the same label. For FLICKR25K and NUS-WIDE, two images are regarded as similar if they have at least one common label. Three evaluation metrics are utilized: Mean Average Precision (MAP), Precision-recall curve, and TopN-precision curve. For FLICKR25K and NUS-WIDE, we adopt *MAP*@5000. For CIFAR-10, we adopt *MAP*@50000.

4.2 Experimental Results

Table 1 shows the MAP results of PURPLE and other baseline methods on three datasets FLICKR25K, CIFAR-10, and NUS-WIDE with hash code lengths varying from 16 to 64. In addition, the Precision-recall curves and the TopN-precision curves of SSDH, CUDH, MLS^{3}RDUH, GLC, and PURPLE on the three datasets are shown in Figure 3 and Figure 4, respestively. Based on the results, it can be observed that:

- Methods related with contrastive learning that utilizes the strong agreement between different views of the same images (SPQ and PURPLE) outperform other baselines, implying that contrastive learning helps generate high-quality hash codes and further enhance the model performance.
- Our method PURPLE outperforms all the competing baselines on all three datasets. Specifically, compared with the representative self-supervised method SPQ, PURPLE achieves an improvement of 16.2%, 5.6% and 2.8% for the average MAP on the dataset CIFAR-10, FLICKR25K and NUS-WIDE, respectively, indicating that hash codes generated by our method can preserve the semantic structure information more efficiently by considering the global semantic structure of both the deep feature space and the hash code space.
- The Precision-recall curves of PURPLE are always on top of the other curves of four baselines and the TopN-precision curves of PURPLE are always above the other curves, demonstrating that PURPLE can achieve superior performances under the Hamming ranking-based evaluation.

4.3 Ablation Study

To investigate the effectiveness of the semantic prototypical consistency learning and the contrastive prototypical consistency learning of PURPLE, we introduce two variants of our method:

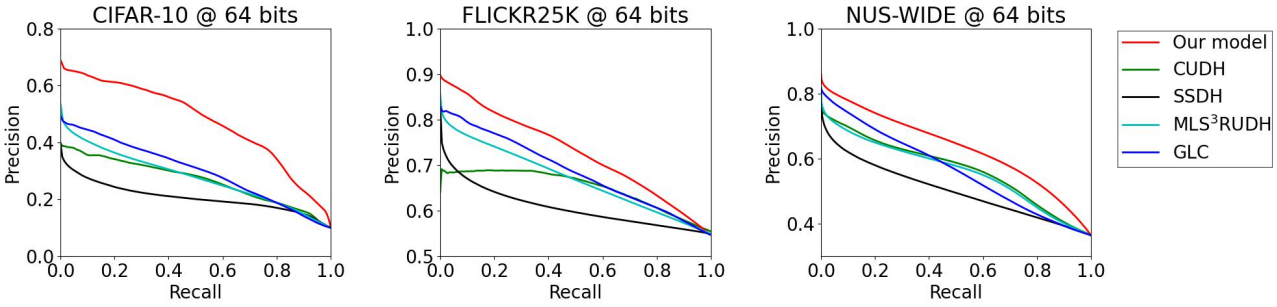

Figure 3: The Precision-recall curves on three benchmark datasets with hash codes @ 64 bits.

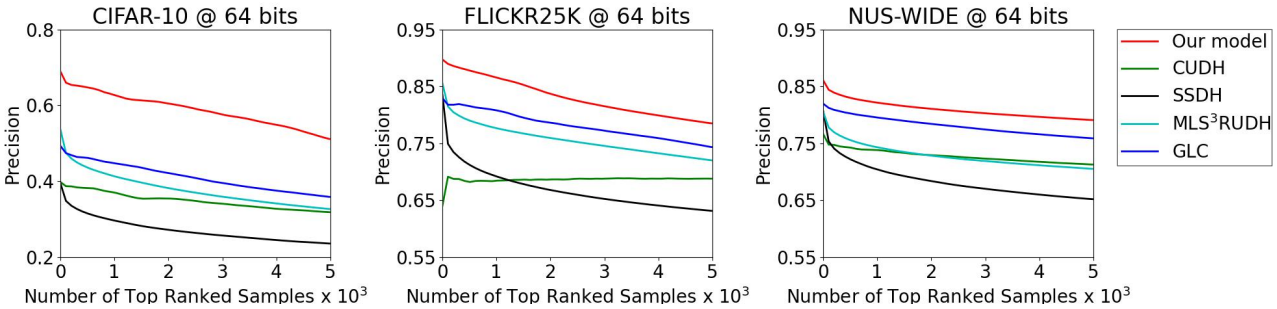

Figure 4: The Top$N$-precision curves on three benchmark datasets with hash codes @ 64 bits.

Table 2: Ablation study on CIFAR-10, FLICKR25K and NUS-WIDE.

Methods	CIFAR-10			FLICKR25K			NUS-WIDE		
	16bits	32bits	64bits	16bits	32bits	64bits	16bits	32bits	64bits
PURPLE-v1	0.482	0.486	0.518	0.788	0.796	0.809	0.772	0.783	0.792
PURPLE-v2	0.395	0.424	0.439	0.762	0.788	0.797	0.738	0.757	0.783
PURPLE	**0.515**	**0.520**	**0.544**	**0.814**	**0.828**	**0.831**	**0.799**	**0.802**	**0.808**

- PURPLE-v1 only resorts to the semantic prototypical consistency learning module, which constructs the pseudo-graph S_f as the guidance of preserving the semantic structure of the whole datasets during the hash code learning process. By comparing the results of PURPLE-v1 and the full model PURPLE in Table 2, it can be observed that the contrastive prototypical consistency learning can further help improve the model performance.
- PURPLE-v2 only performs the contrastive prototypical consistency learning by maximizing the agreement of different views of each image in regards to the hash code prototypical assignments. PURPLE surpasses PURPLE-v2, demonstrating the effectiveness of the semantic prototypical consistency learning in preserving the underlying semantic structure for hash code learning.

4.4 Parameter Sensitivity

We further study the influence of the number of prototypes M and the threshold T. As can be seen in the left column of Figure 5, the performance of our model stabilizes when M is in the range of [30, 120] for both two datasets, and degrades when the number of prototypes is smaller than 30 or larger than 120. Moreover, as can be observed from the right column of Figure 5, PURPLE can achieve more considerable performances when T is in the range of [0.7, 0.9]. Hence, M and T are set to 50 and 0.8 as default, respectively.

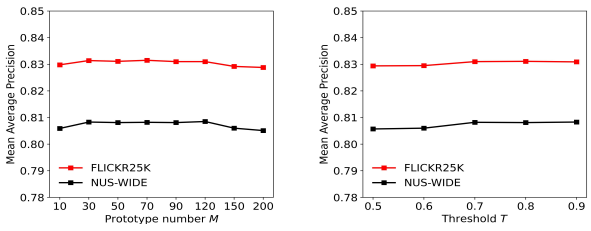

Figure 5: Sensitivity analysis of the number of prototypes M and the threshold parameter T with hash codes @ 64 bits.

4.5 Visualization

Figure 6 demonstrates the visualization results by t-SNE [41] of MLS³RUDH, SPQ and PURPLE following [2, 4]. By comparing the

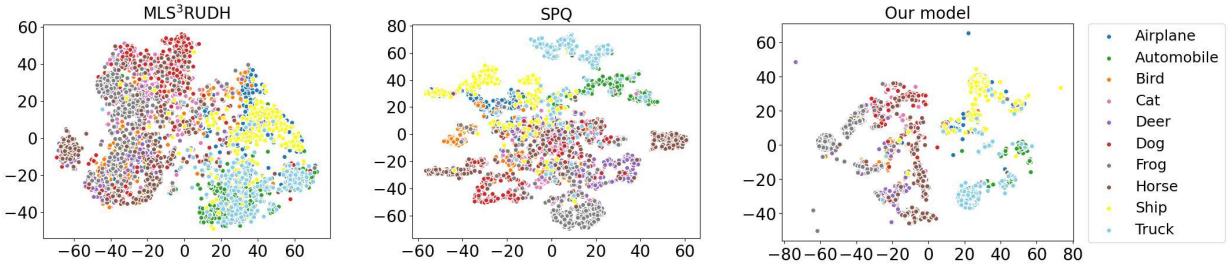

Figure 6: The t-SNE visualization of hash codes @ 64 bits on CIFAR-10.

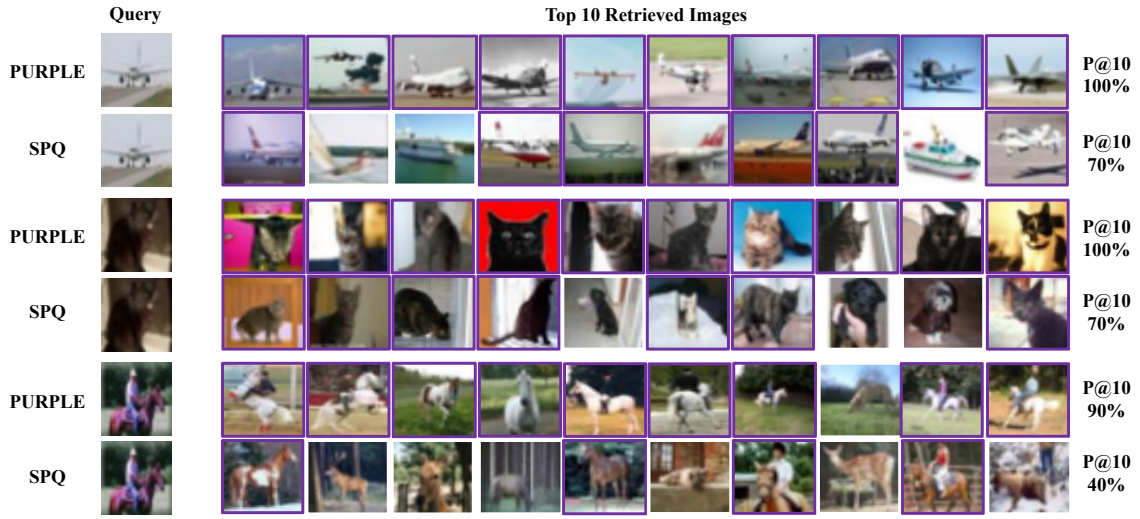

Figure 7: Examples of the top 10 retrieved images and Precision@10 on CIFAR-10 with hash codes @ 32 bits.

results of our model and MLS^{3}RUDH, it can be found that the hash codes in different labels generated by PURPLE are more clearly separated, indicating that the semantic structure information is significantly better preserved in the hash codes generated by PURPLE. In addition, by comparing the results of our model and SPQ, hash codes generated by PURPLE exhibit more discriminative structures. For instance, the blue points (i.e., the category Track) and the yellow points (i.e., the category Ship) are more clearly separated by our model. Furthermore, we show the top 10 retrieved images by PURPLE and SPQ based on 32-bit hash codes in Figure 7, where images in purple boxes are correct results. Benefiting from the semantic prototypical consistency learning and the contrastive prototypical consistency learning of our proposed method PURPLE, it can be observed that our model provides much more relevant image retrieval results.

5 CONCLUSION

This paper studies deep unsupervised hashing, and a novel method called PURPLE is proposed. PURPLE consists of two modules named semantic prototypical consistency learning and contrastive prototypical consistency learning, which explore and preserve the global semantic structure in the deep feature space and the hash code space, respectively. Furthermore, a semantic prototypical consistency loss and a contrastive prototypical consistency loss are proposed to encourage both the semantic structure alignment of two spaces and the clustering consistency for different views in the hash code space. Experiments on a range of benchmarks validate the effectiveness of our PURPLE. By comparing our method with competing baselines, PURPLE surpasses the state-of-the-art unsupervised hashing methods by large margins. In future work, we will explore our methods in various retrieval tasks such as cross-modal unsupervised hashing and semi-supervised hashing.

6 ACKNOWLEDGMENTS

This work was supported in part by the NSFC fund 62176077, in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2019Bl515120055, in part by the Shenzhen Key Technical Project under Grant 2020N046, in part by the Shenzhen Fundamental Research Fund under Grant JCYJ20210324132210025, and in part by the Medical Biometrics Perception and Analysis Engineering Laboratory, Shenzhen, China. This work was also supported by Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies (2022B1212010005).

REFERENCES

[1] Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. 2020. Self-labelling via simultaneous clustering and representation learning. In *ICLR*.

[2] Yue Cao, Mingsheng Long, Bin Liu, and Jianmin Wang. 2018. Deep cauchy hashing for hamming space retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1229–1237.

[3] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Philip S Yu. 2017. Hashnet: Deep learning to hash by continuation. In *Proceedings of the IEEE international conference on computer vision*. 5608–5617.

[4] Zhangjie Cao, Ziping Sun, Mingsheng Long, Jianmin Wang, and Philip S Yu. 2018. Deep priority hashing. In *Proceedings of the 26th ACM international conference on Multimedia*. 1653–1661.

[5] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. 2018. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision*. 132–149.

[6] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. 2020. Unsupervised Learning of Visual Features by Contrasting Cluster Assignments. In *Thirty-fourth Conference on Neural Information Processing Systems (NeurIPS)*.

[7] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709* (2020).

[8] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. 2009. NUS-WIDE: a real-world web image database from National University of Singapore. In *Proceedings of the ACM international conference on image and video retrieval*. 1–9.

[9] Marco Cuturi. 2013. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems* 26 (2013), 2292–2300.

[10] Bo Dai, Ruiqi Guo, Sanjiv Kumar, Niao He, and Le Song. 2017. Stochastic generative hashing. In *International Conference on Machine Learning*. PMLR, 913–922.

[11] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. 2014. Discriminative unsupervised feature learning with convolutional neural networks. *Advances in neural information processing systems* (2014).

[12] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. 1999. Similarity search in high dimensions via hashing. In *Vldb*, Vol. 99. 518–529.

[13] Yifan Gu, Shidong Wang, Haofeng Zhang, Yazhou Yao, and Li Liu. 2019. Clustering-driven unsupervised deep hashing for image retrieval. *Neurocomputing* 368 (2019), 114–123.

[14] Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Vol. 2. 1735–1742.

[15] Zhongkai Hao, Chengqiang Lu, Zhenya Huang, Hao Wang, Zheyuan Hu, Qi Liu, Enhong Chen, and Cheekong Lee. 2020. ASGN: An active semi-supervised graph neural network for molecular property prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

[16] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9729–9738.

[17] Jiabo Huang, Shaogang Gong, and Xiatian Zhu. 2020. Deep semantic clustering by partition confidence maximisation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8849–8858.

[18] Mark J Huiskes and Michael S Lew. 2008. The MIR flickr retrieval evaluation. In *Proceedings of the 1st ACM international conference on Multimedia information retrieval*. 39–43.

[19] Young Kyun Jang and Nam Ik Cho. 2021. Self-supervised product quantization for deep unsupervised image retrieval. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 12085–12094.

[20] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[21] Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan. 2015. Simultaneous feature learning and hash coding with deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3270–3278.

[22] Junnan Li, Caiming Xiong, and Steven CH Hoi. 2021. Comatch: Semi-supervised learning with contrastive graph regularization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 9475–9484.

[23] Qi Li, Zhenan Sun, Ran He, and Tieniu Tan. 2017. Deep supervised discrete hashing. In *Advances in Neural Information Processing Systems*. 2482–2491.

[24] Yingjian Li, Yingnan Gao, Bingzhi Chen, Zheng Zhang, Guangming Lu, and David Zhang. 2021. Self-supervised exclusive-inclusive interactive learning for multi-label facial expression recognition in the wild. *IEEE Transactions on Circuits and Systems for Video Technology* 32, 5 (2021), 3190–3202.

[25] Yunfan Li, Peng Hu, Zitao Liu, Dezhong Peng, Joey Tianyi Zhou, and Xi Peng. 2021. Contrastive clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 8547–8555.

[26] Yingjian Li, Guangming Lu, Jinxing Li, Zheng Zhang, and David Zhang. 2020. Facial Expression Recognition in the Wild Using Multi-level Features and Attention Mechanisms. *IEEE Transactions on Affective Computing* (2020).

[27] Yingjian Li, Zheng Zhang, Bingzhi Chen, Guangming Lu, and David Zhang. 2022. Deep margin-sensitive representation learning for cross-domain facial expression recognition. *IEEE Transactions on Multimedia* (2022).

[28] Kevin Lin, Jiwen Lu, Chu-Song Chen, and Jie Zhou. 2016. Learning compact binary descriptors with unsupervised deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1183–1192.

[29] Mingbao Lin, Rongrong Ji, Hong Liu, and Yongjian Wu. 2018. Supervised online hashing via hadamard codebook learning. In *Proceedings of the 26th ACM international conference on Multimedia*. 1635–1643.

[30] Shuai Lin, Pan Zhou, Zi-Yuan Hu, Shuojia Wang, Ruihui Zhao, Yefeng Zheng, Liang Lin, Eric Xing, and Xiaodan Liang. 2021. Prototypical Graph Contrastive Learning. *arXiv preprint arXiv:2106.09645* (2021).

[31] Xiao Luo, Haixin Wang, Chong Chen, Huasong Zhong, Hao Zhang, Minghua Deng, Jianqiang Huang, and Xian-Sheng Hua. 2022. A Survey on Deep Hashing Methods. *ACM Transactions on Knowledge Discovery from Data* (2022).

[32] Xiao Luo, Daqing Wu, Chong Chen, Jinwen Ma, and Minghua Deng. 2021. Deep Unsupervised Hashing by Global and Local Consistency. In *2021 IEEE International Conference on Multimedia and Expo (ICME)*. 1–6.

[33] Xiao Luo, Daqing Wu, Zeyu Ma, Chong Chen, Minghua Deng, Jianqiang Huang, and Xian-Sheng Hua. 2021. A Statistical Approach to Mining Semantic Similarity for Deep Unsupervised Hashing. In *Proceedings of the 29th ACM International Conference on Multimedia*. 4306–4314.

[34] Xiao Luo, Daqing Wu, Zeyu Ma, Chong Chen, Minghua Deng, Jinwen Ma, Zhongming Jin, Jianqiang Huang, and Xian-Sheng Hua. 2021. Cimon: Towards high-quality hash codes. In *Proceedings of the International Joint Conference on Artificial Intelligence*. 902–908.

[35] Zeyu Ma, Yuhang Guo, Xiao Luo, Chong Chen, Minghua Deng, Wei Cheng, and Guangming Lu. 2022. DHWP: Learning High-Quality Short Hash Codes Via Weight Pruning. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 4783–4787.

[36] Zexuan Qiu, Qinliang Su, Zijing Ou, Jianxing Yu, and Changyou Chen. 2021. Unsupervised Hashing with Contrastive Information Bottleneck. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

[37] David Schneider, Saquib Sarfraz, Alina Roitberg, and Rainer Stiefelhagen. 2022. Pose-based contrastive learning for domain agnostic activity representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3433–3443.

[38] Yuming Shen, Jie Qin, Jiaxin Chen, Mengyang Yu, Li Liu, Fan Zhu, Fumin Shen, and Ling Shao. 2020. Auto-encoding twin-bottleneck hashing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2818–2827.

[39] K. Simonyan and A. Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*.

[40] Rong-Cheng Tu, Xian-Ling Mao, and Wei Wei. 2020. MLS3RDUH: Deep Unsupervised Hashing via Manifold based Local Semantic Similarity Structure Reconstructing. In *Proceedings of the International Joint Conference on Artificial Intelligence*. 3466–3472.

[41] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

[42] Jingdong Wang, Heng Tao Shen, Jingkuan Song, and Jianqiu Ji. 2014. Hashing for similarity search: A survey. *arXiv preprint arXiv:1408.2927* (2014).

[43] Yair Weiss, Antonio Torralba, and Rob Fergus. 2009. Spectral hashing. *Advances in neural information processing systems* (2009).

[44] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. 2018. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3733–3742.

[45] Cheng Yan, Guansong Pang, Xiao Bai, Chunhua Shen, Jun Zhou, and Edwin Hancock. 2019. Deep hashing by discriminating hard examples. In *Proceedings of the 27th ACM International Conference on Multimedia*. 1535–1542.

[46] Erkun Yang, Cheng Deng, Tongliang Liu, Wei Liu, and Dacheng Tao. 2018. Semantic structure-based unsupervised deep hashing. In *Proceedings of the International Joint Conference on Artificial Intelligence*. 1064–1070.

[47] Erkun Yang, Tongliang Liu, Cheng Deng, Wei Liu, and Dacheng Tao. 2019. Distillhash: Unsupervised deep hashing by distilling data pairs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2946–2955.

[48] Li Yuan, Tao Wang, Xiaopeng Zhang, Francis EH Tay, Zequn Jie, Wei Liu, and Jiashi Feng. 2020. Central similarity quantization for efficient image and video retrieval. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3083–3092.

[49] Wanqian Zhang, Dayan Wu, Yu Zhou, Bo Li, Weiping Wang, and Dan Meng. 2020. Deep unsupervised hybrid-similarity hadamard hashing. In *Proceedings of the 28th ACM International Conference on Multimedia*. 3274–3282.

[50] Shu Zhao, Dayan Wu, Wanqian Zhang, Yu Zhou, Bo Li, and Weiping Wang. 2020. Asymmetric Deep Hashing for Efficient Hash Code Compression. In *Proceedings of the 28th ACM International Conference on Multimedia*. 763–771.

[51] Huasong Zhong, Jianlong Wu, Chong Chen, Jianqiang Huang, Minghua Deng, Liqiang Nie, Zhouchen Lin, and Xian-Sheng Hua. 2021. Graph Contrastive Clustering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.