Weighted Contrastive Hashing

Jiaguo Yu, Huming Qiu, Dubing Chen, and Haofeng Zhang (✉)

School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China.
{yujiaguo, 120106222682, db.chen, zhanghf}@njust.edu.cn

Abstract. The development of unsupervised hashing is advanced by the recent popular contrastive learning paradigm. However, previous contrastive learning-based works have been hampered by (1) insufficient data similarity mining based on global-only image representations, and (2) the hash code semantic loss caused by the data augmentation. In this paper, we propose a novel method, namely Weighted Contrative Hashing (WCH), to take a step towards solving these two problems. We introduce a novel mutual attention module to alleviate the problem of information asymmetry in network features caused by the missing image structure during contrative augmentation. Furthermore, we explore the fine-grained semantic relations between images, *i.e.*, we divide the images into multiple patches and calculate similarities between patches. The aggregated weighted similarities, which reflect the deep image relations, are distilled to facilitate the hash codes learning with a distillation loss, so as to obtain better retrieval performance. Extensive experiments show that the proposed WCH significantly outperforms existing unsupervised hashing methods on three benchmark datasets. Code is available at: http://github.com/RosieYuu/WCH.

Keywords: Unsupervised Image Retrieval · Deep Hashing · Contrastive Learning · Mutual Attention · Weighted Similarities.

1 Introduction

With the advancement of deep neural networks, deep hash has become one of the most studied approaches for Approximate Nearest Neighbors (ANN) in large-scale image retrieval. Earlier studies rely heavily on artificial annotations, which makes it difficult to apply in real-world scenarios due to the high labor costs. As a result, unsupervised deep hashing [27,22,23,36] has gradually become the major research direction in this field, with the recent boom in unsupervised learning [3,13,4,26,2,12]. The key difficulty with unsupervised hash is that the ad-hoc encoding process does not extract the key information for hashing, precisely because of the lack of supervised information. Hence, numerous methods have been proposed to learn better discrete representations for hashing in unsupervised setting.

A large family of recent unsupervised hash learning tasks is based on contrastive learning [3,13,4]. These methods build upon instance discrimination,

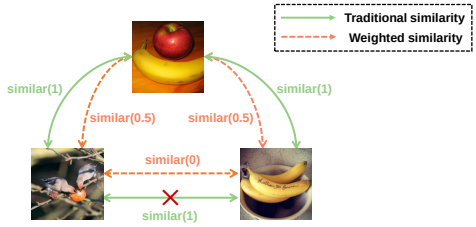

Fig. 1. An example of the conflict of the traditional similarity calculation approach. A typical unsupervised method will treat the top image with both the bottom-left and bottom-right images as similar pairs, because they have a common label. In this case, the two images in the bottom left and bottom right corners should be considered as a similar sample pair. However, the fact is that these two images do not have a common label and they should be considered as a dissimilar pair.

which constructs similar and dissimilar instances and learns the discrete representations by prompting the model to pull in the similar instances and push away the dissimilar instances. With simply the most fundamental concepts for contrastive learning, existing methods[27,23,36] based on contrastive learning have achieved significant success.

Despite their success, most of the current methods mainly focus on adjusting the contrastive loss to fit the hash learning criterion [27,23]. However, directly combining contrastive loss and unsupervised hashing tasks like this leads to two problems. On the one hand, an instance discrimination-based approach leads to the fact that even if the samples are very similar, they still need to be forced apart, *i.e.*, the sample similarity obtained in this way is unreliable. On the other hand, calculating the similarity with the feature vector or hash code of a whole image may lead to the following problem: The top image in Fig. 1 is associated with the labels of apple and banana; the bottom left image is associated with the labels of apple and bird; and the bottom right image is associated with the labels of banana and bowl. In the traditional method, the similarity of both the bottom left image and the bottom right image according to the top image is considered full similar. Therefore, we can say that the bottom left and bottom right images are also very similar. However, the labels of the bottom left and bottom right images do not overlap, *i.e.*, they are actually dissimilar. Based on this, we raise a question: *how to define or even use the similarity between samples to learn high-quality hash codes?*

Curiously, most existing approaches do not focus on this problem. To the best of our knowledge, NSH [36] uses Neural Sorting Operators to obtain the permutation of a vector of similarity scores, and it employs the sorted similarity results to pick the top m positive samples of the anchor, *i.e.*, it improves the comparison by increasing the number of positive samples in the learning framework. However, in the experiment, the optimal number of positive samples is fixed at 3, and all of them are considered fully similar. There are two drawbacks.

First, the anchor image and the augmented similar image, especially processed with random crop, are not always similar, *e.g.*, the cropped image only contains the background, which will prompt the network to learn the background rather than the object representation in the image. Second, for multi-labeled images, the positions and sizes of objects vary greatly, and it is difficult to learn a single depth representation that fits all objects. The quality of the hash codes obtained in this way is not high, which will have an impact on the final retrieval results. This can also explain why NSH [36] boosts highly on single-label datasets, but the boost of MAP on multi-label datasets is not very obvious.

In order to solve the above problem, we propose a novel method called Weighted Contrastive Hashing (WCH) to re-weight the similarity of the anchor image and the others. Concretely, we develop a novel metric rule that is more reasonable and efficient for measuring similar samples, and finally apply this rule to the learning of hash codes for better retrieval performance. We divide each image into a number of patches, and exploit the Vision Transformer (ViT) [7] as the encoder to adapt the patches as the input to the model. To obtain the similar samples of an anchor, we use the aggregated vector of similarity between the patches of different samples as weights. Unlike NSH [36], we do not selectively pick the most relevant samples as the positive samples for contrastive learning, but assign trainable weights to all candidate samples, which represent the degree of similarity between samples. That is, we can consider an image pair as less or more similar, rather than stating them as fully similar or dissimilar in absolute terms. Notably, we demonstrate in the experimental section that our method works better than NSH [36]. In addition, to solve the problem of insufficient similarity between augmented images and anchor images, we propose a Mutual Attention (MA) module to reset the weights of each patch of them by calculating their similarity, which can guarantee sufficient similarity of them to make them the most similar pair, so as to facilitate the hash code learning towards the correct direction. In a nutshell, our main contributions are summarized as follows:

- To the best of our knowledge, this is the first time that weighted contrastive learning has been introduced to image retrieval tasks. It alleviates the problem that certain anchor images and negative samples, which are similar enough for the hashing task, are treated as dissimilar pairs.
- We propose a Mutual Attention module to achieve information complementation between the augmented anchor image and the positive sample, avoiding the lack of key information for hashing.
- The excellent performance of our WCH model is extensively demonstrated by comparing it to 18 state-of-the-art hashing frameworks on three benchmark datasets, *i.e.*, CIFAR-10, NUS-Wide, and MS COCO.

2 Related Works

In this section, we will briefly introduce some unsupervised hashing methods here.

Unsupervised Hashing. Early unsupervised hashing methods mainly focus on projecting images to compact representations by constraining the learned hash codes to fit several principles, *e.g.*, quantization [11], balancing [18]. Several recent works using deep learning pay attention to how to generate high quality hash codes [17,32,23,10]. Some others try to preserve the semantic similarity in hash codes [28,35,30], while the majority of methods adopt image pseudo labels with pre-trained networks to convert the unsupervised hashing to fully a supervised learning [37]. Their performance is usually evaluated against ranked candidates. However, they did not try to sort them during training to mine their similarity.

Hashing with Contrastive learning. Contrastive learning has been a very successful approach for unsupervised hashing tasks. Typical examples include CIMON [23], CIB [27], NSH [36]. All of these methods utilize the contrastive learning framework. As we mentioned before, these methods do not well combine the contrastive learning framework with the hash retrieval task. For example, both CIMON and CIB define the data-enhanced version of an image as a positive sample, and a negative sample is formed by sampling the views of different images. It leads to the possibility that images considered as negative samples may contain positive samples, which will have an impact on the retrieval results. On the other hand, although NSH considers this problem, they simply rank the similarity between anchor samples and select quantitative positive candidates, which does not take into account the similarity degree between the anchor images and augmented images.

Mining Similarity for Unsupervised Hashing. Some methods based on mining similarity aim at solving unsupervised hashing tasks using pairwise methods, *e.g.*, SSDH [34] is a representative method studied in this area. It sets two thresholds at pairwise distances and constructs a similarity structure, and then image features are extracted and hash code learning is performed. However, using two rough thresholds to determine whether they are similar or not is usually unreliable. DistillHash [35] extracts similarity signals using similarity signals from local structures, and further constructs an efficient and adaptive semantic graph, which is updated by decoding it in the context of an autoencoder for hash code learning. MLS^{3}RDUH [33] reconstructs a local semantic similarity structure by exploiting the intrinsic flow structure and cosine similarity in the feature space. DATE [22] improves the commonly used cosine distance by proposing a distribution-based metric. In contrast to these methods, WCH guides the learning of hash codes based on the weighted similarity between patches assigned to each anchor and the rest of the samples.

3 Weighted Contrastive Learning

3.1 Preliminaries

To better explain our method in the next section, we first introduce some concepts and preliminary knowledge here.

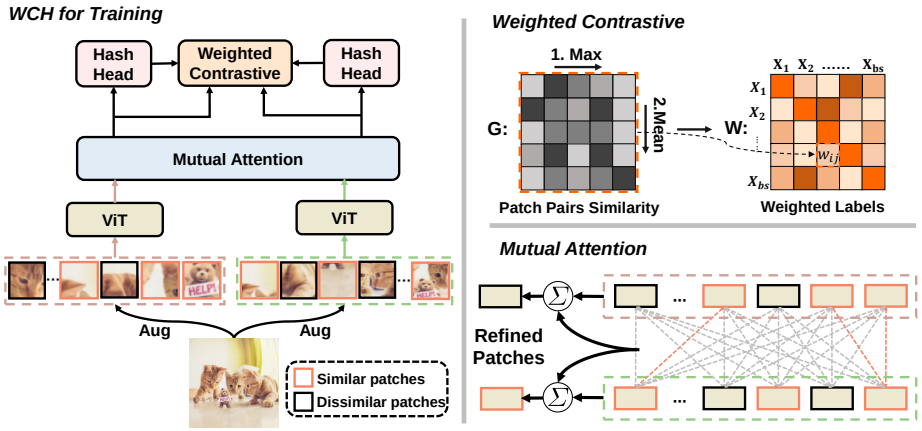

Fig. 2. Overall architecture of the proposed weighted contrastive hashing.

Patch Generation. Following ViT [7], we divide an image $\mathbf{X} \in \mathbb{R}^{s\times s\times c}$ into non-overlapping patches $\mathbf{x}_i \in \mathbb{R}^{p\times p\times c}$, where $i = 1, 2, \cdots, n$. It is obvious that $s^2 = n \times p^2$ and c is the number of image channels.

WCH encoder. Recent work proposes the use of the ViT model as a universal feature extractor [8]. Inspired by these works, we also use ViT as an encoder for our model. We first flatten the patches $\mathbf{x}_i$ into a vector $\mathbf{p}_i \in \mathbb{R}^{1\times d}$, where d is the dimension of the vector, and then use a trainable linear projector **LP** to map the vector to embedding. The output of this projection is referred to the patch embedding as follows:

$$\mathbf{E}_i = \mathbf{LP}(\mathbf{p}_i), \tag{1}$$

where $\mathbf{E}_i$ is the patch embedding associated with the i-th patch. Unlike the standard ViT, our model does not use the class token. We add the position embedding into the patch embeddings, and the final embedding for ViT Input is:

$$\mathbf{PE}_i = \mathbf{E}_i + \mathbf{PoE}, \tag{2}$$

where, **PoE** stands for the position embedding, and **PE** is the final projecting embedding, which will be fed into the transformer encoder $f_\theta(\cdot)$.

Binarization. In WCH, we revisit the problem of how to evaluate whether a candidate sample is comparable to an anchor in the contrastive learning framework and obtain higher quality hash codes in the image retrieval task. As for an image retrieval task, the goal is to learn a binary vector $\mathbf{b}_i \in \{-1, 1\}^l$ by mapping the data $\mathbf{x}_i$ into the encoder, where l is the length of the hash code. In general, the hash code is obtained by the sign function:

$$\mathbf{b}_i = \text{sign}(h(\mathbf{x}_i)) \in \{-1, 1\}^l, \tag{3}$$

where $h(\cdot)$ is the encoding function, which mainly consists of the WCH encoder and a one-layer projector. Since the sign$(\cdot)$ function is non-differentiable,

we adopt a straight-through estimator (STE) [1] that allows back-propagation through $\mathbf{b}_i$.

3.2 Overall Architecture

WCH employs contrastive learning as an unsupervised framework, which typically defines positive pairs as different augmented parts of the same image and negative pairs as samples of different images. Given a batch of N samples $\left\{\mathbf{X}^1, \cdots, \mathbf{X}^i, \cdots, \mathbf{X}^N\right\}$, it first goes through two different data augmenters to get two different views $\tilde{\mathbf{X}}^i$ and $\hat{\mathbf{X}}^i$. Then, we divide each image into n non-overlapping patches, $\tilde{\mathbf{X}}^i = \left[\tilde{\boldsymbol{x}}_1^i; \cdots; \tilde{\boldsymbol{x}}_k^i; \cdots; \tilde{\boldsymbol{x}}_n^i\right], \hat{\mathbf{X}}^i = \left[\hat{\boldsymbol{x}}_1^i; \cdots; \hat{\boldsymbol{x}}_k^i; \cdots; \hat{\boldsymbol{x}}_n^i\right]$. As we described in Sec. 3.1, we employ ViT as the encoder, and feed the patches into it to generate the corresponding encoded features $f_\theta(\tilde{\boldsymbol{x}}_k^i)$ and $f_\theta(\hat{\boldsymbol{x}}_k^i)$.

During traditional augmentation, the augmented two images are usually taken as a similar pair to guide the training direction. However, some cropped images containing background only are totally different from others, which might damage the training process. Therefore, we employ the Mutual Attention (MA) module to re-weight the image patches to guarantee similarity between them. After that, the weighted image similarity is calculated by computing the patch similarity between different images, and it is subsequently used to construct the final weighted contrastive loss function. The overall architecture is illustrated in Fig. 2.

3.3 Mutual Attention

Given any two encoded patches $f_\theta(\tilde{\boldsymbol{x}}_k^i)$ and $f_\theta(\hat{\boldsymbol{x}}_t^i)$ in the corresponding augmented pair, the similarity of them can be calculated as

$$\mathbf{s}_{k,t} = f_\theta(\tilde{\boldsymbol{x}}_k^i)^T f_\theta(\hat{\boldsymbol{x}}_t^i). \tag{4}$$

Therefore, we can construct a similarity matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$, which measures the similarity between each patch of the augmented pair. Then, we normalize the row vectors and column vectors respectively with softmax function:

$$\begin{cases} \mathbf{S}^1 = softmax(\mathbf{s}_{1*}, \cdots, \mathbf{s}_{i*}, \cdots, \mathbf{s}_{n*}) \\ \mathbf{S}^2 = softmax(\mathbf{s}_{*1}, \cdots, \mathbf{s}_{*j}, \cdots, \mathbf{s}_{*n}) \end{cases}, \tag{5}$$

where $\mathbf{s}_{i*}$ stands for the i-th row of $\mathbf{S}$, and $\mathbf{s}_{*j}$ means the j-th column of $\mathbf{S}$. Then, the refined patch vector is reconstructed with the following calculation:

$$\begin{cases} \tilde{\boldsymbol{f}}_k^i = \sum_{j=1}^n s_{j,k}^2 f_\theta(\tilde{\boldsymbol{x}}_j^i) \\ \hat{\boldsymbol{f}}_k^i = \sum_{j=1}^n s_{k,j}^1 f_\theta(\hat{\boldsymbol{x}}_j^i) \end{cases}, \tag{6}$$

where $s_{j,k}^1$ is the j-th row and k-column element of $\mathbf{S}^1$, and $s_{k,j}^2$ is analogously defined. After this operation, the refined augmented pair can be guaranteed to be similar, and thence be undoubtedly treated as a positive pair.

3.4 Weighted Similarities Calculation

Previous works often use embedding vectors to explore the relationship between different images. Specifically, most existing unsupervised hashing methods assume binary similarity between two images, *i.e.*, two images can be similar (positive sample) or dissimilar (negative sample). For example, NSH [36] uses hash codes to calculate the degree of similarity between images, and then ranks them according to the magnitude of similarity, and selects the top m positive samples according to the result of the ranking, *i.e.*, determines that these m samples and anchor are similar. However, selecting positive samples based on similarity like this will cause two problems. First, there may be noise in the positive samples. Since the number of positive samples is set to a fixed value, forcing a fixed number of positive samples based on the ranking results will slow down the convergence of the model. Second, the results of the first m closest samples may not be equivalent. For an anchor image $\mathbf{X}_i$, $\mathbf{X}_j$ is one of the selected m positive samples that are similar to $\mathbf{X}_i$ in one iteration. However, it is possible that in another iteration, $\mathbf{X}_j$ is not one of the m closest samples since there are more similar images in this training batch. In this situation, $\mathbf{X}_j$ will be treated as a negative sample of $\mathbf{X}_i$, which results in inconsistency with the former one, and this conflict will damage the training process and cause the training to fail to converge.

Weighted Labels Processing. To tackle these problems, instead of adopting a strategy such as selecting positive samples, we reformulate the rules for computing the similarity between images and use the obtained similarity to re-weight the contrastive loss to capture the semantic information that may overlap between the anchor and negative samples. In WCH, we exploit the fine-grained interaction results between patches to explore the relationship between different images.

Specifically, suppose there are two patch features $\boldsymbol{f}_k^i$ and $\boldsymbol{f}_t^j$ extracted from two different images $\mathbf{X}^i$ and $\mathbf{X}^j$, respectively. The similarity between them can be defined as

$$g_{kt}^{ij} = (\boldsymbol{f}_k^i)^T \boldsymbol{f}_t^j. \tag{7}$$

Therefore, the similarity matrix of $\mathbf{X}^i$ and $\mathbf{X}^j$ can be constructed as $\mathbf{G}^{ij} \in \mathbb{R}^{n \times n}$. For each row in $\mathbf{G}^{ij}$, the max value represents the most similar batches in $\mathbf{X}^i$ and all patches in $\mathbf{X}^j$, and the mean of the max values of each row is the similarity of $\mathbf{X}^i$ and $\mathbf{X}^j$:

$$w_{ij} = \text{mean}(\max_{row}(\mathbf{G}^{ij})), \tag{8}$$

where $\max_{row}(\cdot)$ means to take the maximum value according to the row direction, and $mean(\cdot)$ stands for calculating the mean value of the vector. For a mini-batch containing bs images, including one augmented image and $bs-1$ other images, the similarity matrix $\mathbf{W} \in \mathbb{R}^{bs \times bs}$ can be constructed with Eq. 8. To fit the value of $\mathbf{W}$ within a proper range, we conduct a temperature weighted row softmax as $\mathbf{W}_{i*} = softmax(\mathbf{W}_{i*}/\tau_w)$, where τ_w [14] is the temperature

coefficient. Furthermore, to guarantee that the augmented images are equivalent to the anchor images, we divide each row with the element w_{ii}:

$$\mathbf{W} = \text{diag}(\text{diag}(\mathbf{W}))^{-1}\mathbf{W}, \tag{9}$$

where $\text{diag}(\cdot)$ means extracting the diagonal vector from a matrix or constructing a diagonal matrix with a vector.

3.5 Training and Inference

For training, we use the maximum similarity between patches to guide the contrastive objective [25]:

$$\mathcal{L}_{\text{WCE}} = -\sum_{i=1}^{bs}\sum_{j=1}^{bs} w_{ij} \log \frac{\exp\left(\tilde{\mathbf{b}}_i \hat{\mathbf{b}}_j^{\intercal}/l/\tau\right)}{\sum_{k=1}^{bs}\exp\left(\tilde{\mathbf{b}}_i \hat{\mathbf{b}}_k^{\intercal}/l/\tau\right)}, \tag{10}$$

where τ is the temperature scale. Finally, the loss function is formulated as

$$\mathcal{L}_{\text{WCH}} = \mathcal{L}_{\text{WCE}} + \mathcal{L}_R, \tag{11}$$

where $\mathcal{L}_R$ refers to the quantization loss and bit balancing loss [9]. The whole learning procedure is shown in Alg. 1.

Inference process. In the inference process, WCH abandons the MA and weighted labeling modules dedicated to training and keeps only the encoder and hash head for generating hash codes characterizing the semantic information of the images. The Hamming distance between the hash codes of the images is then computed to accomplish the retrieval task.

4 Discussion

Remark 1: Why Do We Choose the ViT Encoder? In WCH, our key idea is to use the patch-level semantic information captured by ViT [7] as a benchmark to measure the degree of similarity between arbitrary images and assign corresponding weights to each pair of images by aggregating the similarity between patches to measure the degree of similarity. Unlike the recent self-supervised visual representation learning-based approaches [27,36], they only determine similar samples by the global feature similarity of the whole image. Instead, we introduce a novel inter-patch-based fine-grained interaction module using the ViT model, enabling fine-grained interactions between patches and each pair of images to mine more detailed semantic alignment.

Furthermore, we use the ViT model to address the problems posed by contrastive learning methods that rely on instance discrimination tasks. As mentioned before, positive native pairs are defined as different views of the same image, while negative pairs are formed by sampling views of different images. This common approach ignores their semantic content. Our approach, on the

Algorithm 1: The Training Procedure of WCH.

Input: Dataset $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^{N}$ and batch size n.
Output: Network parameters θ.
for batch in $\mathcal{D}.repeat()$ **do**

```
batch1, batch2 = aug(batch), aug(batch)
f1, f2 = Mθ(batch1), Mθ(batch2)
# mutual attention
sim = einsum('nid, njd → nij', f1, f2)
f1 = einsum('nid, ndj → nij', softmax(sim.T), f1)
f2 = einsum('nid, ndj → nij', softmax(sim), f2)
# weighted similarities calculation
sim = einsum('nid, mjd → nmij', f1, f2)
sim = softmax(sim.max(−1).mean(−1)/τw)
weighted = matmul(diag(diag(sim))^−1, sim)
# hashing
b1, b2 = hash(f1.mean(1)), hash(f2.mean(1))
logits = softmax(matmul(b1, b2.T)/l/τ)
# weighted corss entropy
loss= cross_entropy(logits, weighted)
loss.backward()
```

end

other hand, fully exploits the semantic content of the images and makes reasonable use of fine-grained interaction results as a measure of similarity between images, as detailed in Sec. 3.4.

Remark 2: Why Mutual Attention Helps? First, note the phenomenon that most models construct positive and negative samples by treating the same images produced by different augmenters as positive pairs, while the rest of the samples are considered as negative pairs. However, this manually designed approach involves many manual choices, and inappropriate data augmentation schemes may severely alter the image structure, resulting in data-enhanced images that do not possess label-preserving properties, *i.e.*, images undergo transformations that may lose high-level semantic information. For example, a common data augmentation scheme is random cropping, which may randomly crop out the sample information that contains label-related information for single-labeled images. Similarly, for multi-labeled images, where the position and size of objects vary greatly, the random cropping method will most likely crop out some objects in multi-labeled images, making the sample information contained in multi-labeled images reduced. This operation will lead to asymmetric semantic information between the anchor and the positive samples.

The mutual attention module in Fig. 2 reconstructs the feature vectors associated with the pictures based on the similarity between positive sample pairs of patches. Therefore, it can be seen as a specific attention mechanism. Intuitively, it focuses our attention on the degree of similarity of patch pairs. The attention fraction is used so that the feature vectors of each patch carry information about

Table 1. Performance comparison (mAP) of WCH and the state-of-the-art **unsupervised** hashing methods. *Note that we use a more common setting on NUS-WIDE with the 21 most frequent classes, while some papers report results on 10 classes.

Method	Reference	CIFAR-10			NUS-WIDE			MS COCO		
		16 bits	32 bits	64 bits	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
AGH [21]	ICML11	0.333	0.357	0.358	0.592	0.615	0.616	0.596	0.625	0.631
ITQ [11]	PAMI13	0.305	0.325	0.349	0.627	0.645	0.664	0.598	0.624	0.648
DGH [20]	NeurIPS14	0.335	0.353	0.361	0.572	0.607	0.627	0.613	0.631	0.638
SGH [6]	ICML17	0.435	0.437	0.433	0.593	0.590	0.607	0.594	0.610	0.618
BGAN [31]	AAAI18	0.525	0.531	0.562	0.684	0.714	0.730	0.645	0.682	0.707
BinGAN [38]	NeurIPS18	0.476	0.512	0.520	0.654	0.709	0.713	0.651	0.673	0.696
GreedyHash [32]	NeurIPS18	0.448	0.473	0.501	0.633	0.691	0.731	0.582	0.668	0.710
HashGAN [10]	CVPR18	0.447	0.463	0.481	-	-	-	-	-	-
DVB [29]	IJCV19	0.403	0.422	0.446	0.604	0.632	0.665	0.570	0.629	0.623
DistillHash [35]	CVPR19	0.284	0.285	0.288	0.667	0.675	0.677	-	-	-
TBH [30]	CVPR20	0.532	0.573	0.578	0.717	0.725	0.735	0.706	0.735	0.722
MLS^{3}RDUH [33]	IJCAI20	0.369	0.394	0.412	0.713	0.727	0.750	0.607	0.622	0.641
DATE [22]	MM21	0.577	0.629	0.647	0.793	0.809	0.815	-	-	-
MBE [17]	AAAI21	0.561	0.576	0.595	0.651	0.663	0.673	-	-	-
CIMON [23]*	IJCAI21	0.451	0.472	0.494	-	-	-	-	-	-
CIBHash [27]	IJCAI21	0.590	0.622	0.641	0.790	0.807	0.815	0.737	0.760	0.775
SPQ [15]	ICCV21	0.768	0.793	0.812	0.766	0.774	0.785	-	-	-
NSH [36]	IJCAI22	0.706	0.733	0.756	0.758	0.811	0.824	0.746	0.774	0.783
WCH	**Proposed**	**0.897**	**0.910**	**0.932**	**0.799**	**0.823**	**0.838**	**0.776**	**0.808**	**0.834**

other patches to different degrees. More specifically, this attention mechanism is very useful for multiple patches, especially when there are many classes of objects and the positions are highly variable.

Remark 3: Why Do We Gather W in This Way? The purpose of Weighted Labels is to use the maximum similarity between patches to guide the contrastive objective. Using the maximum similarity between patches in Eq. 8, we can get the most similar patch pair among all patches in the two images. Then the sum of the maximum similarity is averaged. The model learns the fine-grained semantic alignment between patches by applying the weighted label to the contrastive loss.

5 Experiments

In this section, we conduct experiments on three datasets, including one single-labeled dataset and two multi-labeled datasets, to evaluate our method.

5.1 Datasets and Evaluation Metrics

Three benchmark datasets are used in our experiments. **CIFAR-10** [16] consists of 60,000 images from 10 classes. We follow the common setting [10] and select 10,000 images (1000 per class) as the query set. The remaining 50,000 images

are regarded as the database. **NUS-WIDE** [5] has of 81 categories of images. We adopt the 21-class subset following [36]. 100 images of each class are utilized as a query set, with the remaining being the gallery. **MS COCO** [19] is a benchmark for multiple tasks. We use the conventional set with 12,2218 images. We randomly select 5,000 images as queries with the remaining ones the database.

Evaluation Metric. To compare the proposed method with the baselines, we adopt several widely-used evaluation metrics, including the mean average precision (mAP), top-K precision (P@K), precision-recall (PR) curves [37].

5.2 Implementation Details

For all three datasets, the images were resized to $224 \times 224 \times 3$ and we adopt the image augmentation strategies of [3]. The standard ViT-Base [7] was used as the backbone, with patches of size and number 16 and 196, respectively. As in previous work [23,27], we loaded a pre-trained model trained on ImageNet to accelerate the convergence. We used the cosine decay method and trained 50 epochs for all models, with the initial learning rate set to 1×10^{-5}.

5.3 Comparison with the SotA

Baselines. We compare WCH against 18 state-of-the-art baselines, including 3 traditional unsupervised hashing methods and 15 recent unsupervised hashing methods. For fair comparisons, all the methods are reported with identical training and test sets. Additionally, the shallow methods are evaluated with the same deep features as the ones we are using.

Results. Tab. 1 shows the retrieval performance in mAP and Tab. 2 demonstrates the precision of the first 1000 returned images. It can be clearly observed that WCH obtains the best results on all three datasets for the two metrics. Another interesting observation is that WCH significantly outperforms the previous works CIBHash and NSH on different hash bits and datasets. Note that all three methods use contrastive learning. In addition, the P-R curves of WCH and several baselines on CIFAR-10 and MS COCO are reported in Fig. 3, from which it can also be discovered that the curves of our method are highly above those of other methods for all three different code lengths.

5.4 Ablation Studies

In this subsection, we considered the following ablation experiments to verify the effectiveness and contribution of each component of WCH, and the specific results are shown in Tab. 3.

(i) **ViT Baseline.** We first investigate the enhancements that the ViT backbone brings to the unsupervised hashing domain. In this baseline, the class token covering global features is applied directly to the hash head to generate a hash code characterizing the image. Subsequent contrastive loss is used to update the network parameters, which form a design close to the CIB [27] except that

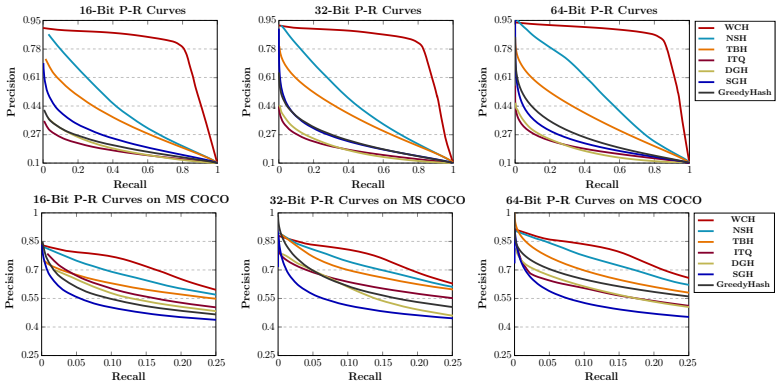

Fig. 3. P-R curves comparison with other methods on CIFAR-10 and MS COCO.

Table 2. P@1000 results on CIFAR-10 and MS COCO.

Method	CIFAR-10			MS COCO		
	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
AGH	0.306	0.321	0.317	0.602	0.635	0.644
DGH	0.315	0.323	0.324	0.623	0.642	0.650
HashGAN	0.418	0.436	0.455	-	-	-
GreedyHash	0.322	0.403	0.444	0.603	0.624	0.675
TBH	0.497	0.524	0.529	0.646	0.698	0.701
CIBHash	0.526	0.570	0.583	0.734	0.767	0.785
NSH	0.691	0.716	0.744	0.733	0.770	0.805
WCH	**0.889**	**0.902**	**0.923**	**0.795**	**0.830**	**0.855**

the network backbone differs. Regrettably, the application of the ViT backbone alone is not sufficient to improve the performance of the unsupervised hashing.
(ii) **Without $\mathcal{L}_R$.** We also reveal the impact of traditional quantization loss and bit balance loss [9] on WCH. It can be seen that these conventional regularizers have no significant improvement in the encoding quality. As a result, we can attribute the good performance entirely to our design.
(iii) **MA $\rightarrow$ mean.** We use this baseline to demonstrate the validity of our MA module. Here we remove the mutual attention mechanism of anchor and positive samples in Eq. 6 and replace it with the averaging operation. Although it also achieves trivially good results, there is still a noticeable margin of difference with the performance of WCH, which indicates that the motivation of mutual attention can play a positive role.
(iv) **Weighted $\rightarrow$ hard.** This baseline does not use weighted labels, but rather the most fundamental hard labels, which means that the weighted contrastive learning degrades to the standard contrastive learning. The non-negligible performance degradation in Tab. 3 precisely illustrates the shortcoming of standard contrastive learning, which cannot close the distance between the anchor and

Table 3. Ablation study results of mAP@1000 on MS COCO. The baselines are constructed by replacing some key modules of WCH.

Baseline	16 bits	32 bits	64 bits
(i) ViT Baseline	0.573	0.595	0.622
(ii) Without $\mathcal{L}_R$	0.773	0.810	0.828
(iii) MA $\rightarrow$ mean	0.742	0.782	0.805
(iv) weighted $\rightarrow$ hard	0.738	0.777	0.799
(v) Without scale	0.461	0.479	0.491
WCH	**0.776**	**0.808**	**0.834**

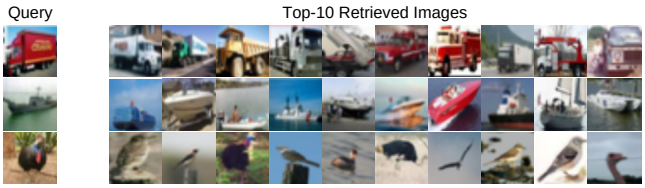

Fig. 4. Examples of top-10 retrieved results of 32-bit on CIFAR-10.

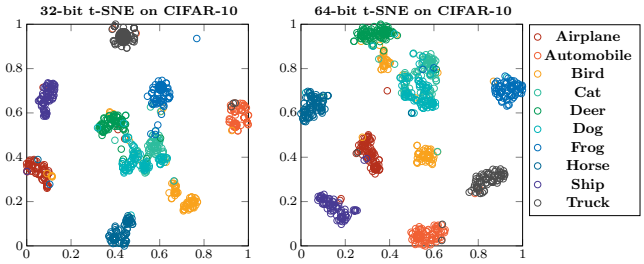

Fig. 5. 32-bit and 64-bit t-SNE visualization results on CIFAR-10.

similar negative samples in the feature space. This also highlights the crucial role played by our core motivation from the side.

(v) **Without scale.** In this baseline, we remove the operation defined in Eq. 9 and simply use the similarity matrix $\mathbf{W}$ in Eq. 8 as a weighted label during the calculation of the loss. We can strikingly see an unexpectedly dramatic performance slippage. Hence, affine mapping based on positive sample similarity is a key factor to guarantee the effectiveness of weighted comparison learning.

Results. Baseline (i) contradicts our intuition that directly replacing the backbone network with ViT can not bring meaningful performance improvement. Baseline (iii) shows that MA is an effective solution to deal with the information asymmetry problem for positive samples. We use baseline (iv) to validate our core motivation that weighted contrast learning can substantially alleviate the class collision problem of negative samples and thus further improve the retrieval performance.

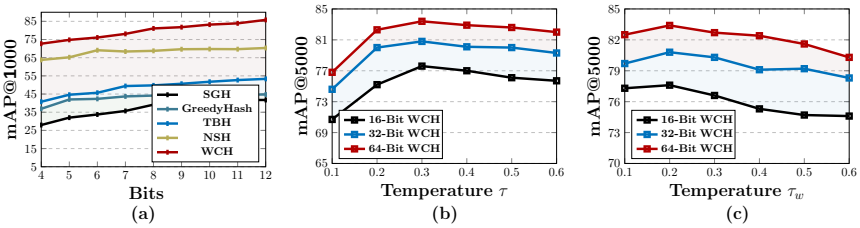

Fig. 6. **(a)** mAP@1000 results with extremely short code lengths on CIFAR-10. **(b)**&**(c)**Effects of different temperatures τ and τ_w on MS COCO.

5.5 Visualization and Hyper-parameters

Visualization. To more intuitively demonstrate the performance of our method, we show the retrieved top-10 images on CIFAR-10 in Fig. 4, where a high semantic accuracy can be observed from the results. In addition, to show whether the embedded hash codes are discriminative enough for retrieval, the t-SNE plots [24] of hash codes for both 32-bit and 64-bit on CIFAR-10 are also illustrated in Fig. 5, where the plotted dots of different classes show obvious boundaries between them, which means that the generated codes are separable and shows the consistency with other results.

Hyper-parameters. In Fig. 6(a), we show the results for very short hash code lengths on CIFAR10. Although the performance varies slightly depending on the hyperparameter settings, it is generally stable and state-of-the-art. We also evaluated the impact of the temperature coefficient τ of the WCE loss and the temperature coefficient τ_w of computing the weighted labels on the final performance of MSCOCO, and we depict these trends in Fig. 6(b) and (c).

6 Conclusion

In this paper, we propose a weighted contrastive hashing model to explore semantic information based on fine-grained information interactions between patches for image retrieval. The proposed mutual attention module can well solve the inconsistency of the anchor image and the augmented images. A weighted coefficient is calculated to weigh the similarities of the images in a training batch, and it can better improve the hash code learning. Extensive experiments show that the proposed method improves the state-of-the-art unsupervised hashing scheme in image retrieval.

Acknowledgements This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grants No. 61872187, and No. 62072246, in part by the Natural Science Foundation of Jiangsu Province under Grant No. BK20201306, and in part by the "111" Program under Grant No. B13022.

References

1. Bengio, Y., Léonard, N., Courville, A.: Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv preprint arXiv:1308.3432 (2013)
2. Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., Joulin, A.: Unsupervised learning of visual features by contrasting cluster assignments. In: NeurIPS (2020)
3. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: ICML (2020)
4. Chen, X., Fan, H., Girshick, R., He, K.: Improved baselines with momentum contrastive learning. arXiv preprint arXiv:2003.04297 (2020)
5. Chua, T.S., Tang, J., Hong, R., Li, H., Luo, Z., Zheng, Y.: Nus-wide: a real-world web image database from national university of singapore. In: CIVR (2009)
6. Dai, B., Guo, R., Kumar, S., He, N., Song, L.: Stochastic generative hashing. In: ICML (2017)
7. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. ArXiv **abs/2010.11929** (2021)
8. Dubey, S.R., Singh, S.K., Chu, W.: Vision transformer hashing for image retrieval. ArXiv **abs/2109.12564** (2021)
9. Erin Liong, V., Lu, J., Wang, G., Moulin, P., Zhou, J.: Deep hashing for compact binary codes learning. In: CVPR (2015)
10. Ghasedi Dizaji, K., Zheng, F., Sadoughi, N., Yang, Y., Deng, C., Huang, H.: Unsupervised deep generative adversarial hashing network. In: CVPR (2018)
11. Gong, Y., Lazebnik, S., Gordo, A., Perronnin, F.: Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. IEEE Transactions on Pattern Analysis and Machine Intelligence **35**(12), 2916–2929 (2013)
12. Grill, J.B., Strub, F., Altch'e, F., Tallec, C., Richemond, P.H., Buchatskaya, E., Doersch, C., Pires, B.Á., Guo, Z.D., Azar, M.G., Piot, B., Kavukcuoglu, K., Munos, R., Valko, M.: Bootstrap your own latent: A new approach to self-supervised learning. ArXiv **abs/2006.07733** (2020)
13. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. In: CVPR (2020)
14. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
15. Jang, Y.K., Cho, N.I.: Self-supervised product quantization for deep unsupervised image retrieval. 2021 IEEE/CVF International Conference on Computer Vision (ICCV) pp. 12065–12074 (2021)
16. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Technical Report, University of Toronto (2009)
17. qiang Li, Y., van Gemert, J.C.: Deep unsupervised image hashing by maximizing bit entropy. In: AAAI (2021)
18. Lin, K., Lu, J., Chen, C.S., Zhou, J.: Learning compact binary descriptors with unsupervised deep neural networks. In: CVPR (2016)
19. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: ECCV (2014)
20. Liu, W., Mu, C., Kumar, S., Chang, S.F.: Discrete graph hashing. In: NeurIPS (2014)

21. Liu, W., Wang, J., Kumar, S., Chang, S.F.: Hashing with graphs. In: ICML (2011)
22. Luo, X., Wu, D., Ma, Z., Chen, C., Deng, M., Huang, J., Hua, X.S.: A statistical approach to mining semantic similarity for deep unsupervised hashing. In: MM (2021)
23. Luo, X., Wu, D., Ma, Z., Chen, C., Zhong, H., Deng, M., Huang, J., Hua, X.s.: Cimon: Towards high-quality hash codes. In: IJCAI (2021)
24. Maaten, L.v.d., Hinton, G.: Visualizing data using t-sne. Journal of Machine Learning Research **9**(Nov), 2579–2605 (2008)
25. Oord, A.v.d., Li, Y., Vinyals, O.: Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748 (2018)
26. Qi, Y., Song, Y.Z., Zhang, H., Liu, J.: Sketch-based image retrieval via siamese convolutional neural network. In: IEEE International Conference on Image Processing (ICIP) (2016)
27. Qiu, Z., Su, Q., Ou, Z., Yu, J., Chen, C.: Unsupervised hashing with contrastive information bottleneck. In: IJCAI (2021)
28. Shen, F., Xu, Y., Liu, L., Yang, Y., Huang, Z., Shen, H.T.: Unsupervised deep hashing with similarity-adaptive and discrete optimization. IEEE Transactions on Pattern Analysis and Machine Intelligence **40**(12), 3034–3044 (2018)
29. Shen, Y., Liu, L., Shao, L.: Unsupervised binary representation learning with deep variational networks. International Journal of Computer Vision **127**(11-12), 1614–1628 (2019)
30. Shen, Y., Qin, J., Chen, J., Yu, M., Liu, L., Zhu, F., Shen, F., Shao, L.: Auto-encoding twin-bottleneck hashing. In: CVPR (2020)
31. Song, J., He, T., Gao, L., Xu, X., Hanjalic, A., Shen, H.T.: Binary generative adversarial networks for image retrieval. In: AAAI (2018)
32. Su, S., Zhang, C., Han, K., Tian, Y.: Greedy hash: Towards fast optimization for accurate hash coding in cnn. In: NeurIPS (2018)
33. Tu, R.C., Mao, X.L., Wei, W.: Mls3rduh: Deep unsupervised hashing via manifold based local semantic similarity structure reconstructing. In: IJCAI (2020)
34. Yang, E., Deng, C., Liu, T., Liu, W., Tao, D.: Semantic structure-based unsupervised deep hashing. In: IJCAI (2018)
35. Yang, E., Liu, T., Deng, C., Liu, W., Tao, D.: Distillhash: Unsupervised deep hashing by distilling data pairs. In: CVPR (2019)
36. Yu, J., Shen, Y., Zhang, H., Torr, P.H.S., Wang, M.: Learning to hash naturally sorts. In: IJCAI (2022)
37. Zhang, H., Liu, L., Long, Y., Shao, L.: Unsupervised deep hashing with pseudo labels for scalable image retrieval. IEEE Transactions on Image Processing **27**(4), 1626–1638 (2018)
38. Zieba, M., Semberecki, P., El-Gaaly, T., Trzcinski, T.: Bingan: Learning compact binary descriptors with a regularized gan. In: NeurIPS (2018)