Unsupervised Hashing with Similarity Distribution Calibration

Kam Woh Ng[1,2]
kamwoh.ng@surrey.ac.uk
Xiatian Zhu[1,3]
xiatian.zhu@surrey.ac.uk
Jiun Tian Hoe*[4]
jiuntian001@e.ntu.edu.sg
Chee Seng Chan[5]
cs.chan@um.edu.my
Tianyu Zhang[6]
macwish@hotmail.com
Yi-Zhe Song[1,2]
y.song@surrey.ac.uk
Tao Xiang[1,2]
t.xiang@surrey.ac.uk

[1] CVSSP, University of Surrey
[2] iFlyTek-Surrey Joint Research Centre on Artificial Intelligence, University of Surrey
[3] Surrey Institute for People-Centred Artificial Intelligence, University of Surrey
[4] Nanyang Technological University
[5] CISiP, Universiti Malaya
[6] GAC R&D Center

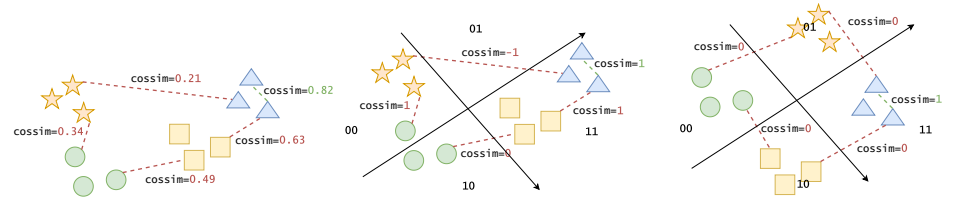
Abstract

Unsupervised hashing methods typically aim to preserve the similarity between data points in a feature space by mapping them to binary hash codes. However, these methods often overlook the fact that the similarity between data points in the continuous feature space may not be preserved in the discrete hash code space, due to the limited similarity range of hash codes. The similarity range is bounded by the code length and can lead to a problem known as *similarity collapse*. That is, the positive and negative pairs of data points become less distinguishable from each other in the hash space. To alleviate this problem, in this paper a novel **Similarity Distribution Calibration** (SDC) method is introduced. SDC aligns the hash code similarity distribution towards a calibration distribution (*e.g.*, beta distribution) with sufficient spread across the entire similarity range, thus alleviating the similarity collapse problem. Extensive experiments show that our SDC outperforms significantly the state-of-the-art alternatives on coarse category-level and instance-level image retrieval. Code is available at https://github.com/kamwoh/sdc.

1 Introduction

Hashing has been used extensively in real-world large-scale image retrieval systems. By converting continuous feature vectors into binary/discrete hash codes for indexing, hashing significantly reduces both computational cost and memory footprint. Recent deep supervised

* The work was done while in Universiti Malaya.



(a) (b) (c)

Figure 1: **(a)** In the original feature space (before hashing), the cosine similarity values are continuous within the range of [-1, 1]. **(b)** After mapping to the hash code space with the conventional feature similarity preservation strategy, the similarity collapse phenomenon emerges due to the limited similarity range (only $K+1$ choices with K the hash code length) in the Hamming space and the original similarity bias (*e.g.*, negative pairs taking moderately positive similarity). **(c)** This problem can be well alleviated by our *Similarity Distribution Calibration* method even with limited cosine similarity values. See Fig. 7 for the result on actual data.

learning to hash [6, 14, 33, 37, 40, 57, 64, 69] have greatly outperformed conventional methods [16, 22, 29, 31, 59]. However, supervised hashing is limited in scalability due to its reliance on a large quantity of labeled training data. A natural solution is to use *unsupervised hashing methods* instead, which do not require costly training data annotation.

The current state-of-the-art unsupervised hashing methods [35, 57] are either based on *preserving individual pairwise similarities* between continuous feature vectors in the learned Hamming space [20, 31]. Compared to the alternative strategies (*e.g.*, reconstruction [7], clustering [34], pseudo-labels [63], and contrastive learning [48, 55, 66]), pairwise similarity preservation is both easier to implement and more efficient, hence advantageous for large-scale applications [25].

However, we reveal that these similarity preservation-based hashing methods suffer from a ***similarity collapse*** problem (see Fig. 1b). That is, the hash code similarities of positive and negative pairs become inseparable. There are two causes: (i) The similarity distribution in the original continuous feature space is biased. In particular, most negative pairs take moderately positive similarity scores (Fig. 1a). (ii) The inherent difference in the ability of continuous feature space and discrete hash code space to accurately preserve similarity between data points. Specifically, the similarities between any two hash codes are of a fixed set of values determined by the code length (*i.e.*, limited capacity), whilst the original feature similarities are continuous (*i.e.*, unlimited capacity). With the limited range of similarity scores in the hash code space, the hashing process is given little chance to recover from the collapsing positive and negative feature similarity scores in the Hamming space (Fig. 1b), resulting in inferior retrieval results.

To alleviate this similarity collapse problem, in this work a novel *Similarity Distribution Calibration* (**SDC**) method is introduced. Instead of preserving the original pairwise similarity scores individually, we regularize the hash code similarity distribution as a whole against a pre-defined calibration distribution (*e.g.*, beta distribution) with sufficient capacity range. Due to this stretching effect, the learned Hamming space is no longer restricted severely by the original biased similarity distribution as in the existing methods. This enables the limited similarity range of Hamming space to be better leveraged, resulting in improved performance (Fig. 1c).

We make the following **contributions**: (i) We reveal the fundamental similarity collapse

problem suffered by existing pairwise similarity preservation-based unsupervised hashing methods. (ii) To address this problem, we propose a Similarity Distribution Calibration (SDC) method by alleviating the severe restriction imposed by the original biased similarity scores. (iii) Extensive experiments validate the superiority of our SDC over state-of-the-art alternatives on four category-level and three instance-level image retrieval benchmarks.

2 Related Work

Although earlier hashing methods [15, 16, 22, 26, 29, 41, 45, 59] are easy to apply in practice, their performance is typically inferior to more recent deep learning counterparts. Deep supervised hashing methods [6, 14, 33, 37, 38, 57, 58, 64, 69] usually achieve better performance over unsupervised ones by using additionally the semantic class labels. However, they are limited in scalability as class label annotation is costly and even impossible in extreme cases (*e.g.*, rare objects). Without this constraint, unsupervised methods are thus more scalable. Existing unsupervised hashing methods can be categorized into the following groups: similarity preservation [20, 21, 24, 31, 32, 35, 36, 42, 52, 57, 71], generative model [5, 11, 56, 73], reconstruction [7, 8, 53, 54], pseudo-labeling [17, 44, 61, 62, 70, 72], clustering [34, 68, 70, 72] and contrastive learning [34, 48, 67].

Among these, similarity preservation-based unsupervised hashing methods achieve the current state-of-the-art performance in image retrieval. They are also simple in design and efficient computationally. The general idea is to learn a set of hash codes based on the feature similarity information. This can be achieved either with projection [16, 26], inferring pseudo-labels [44, 61, 62] or direct preserving the feature similarity [20, 31, 35, 57]. Recent contrastive learning based hashing [44, 48, 65, 66] further push the performance. Beyond all these learning approaches, we focus on the similarity collapse problem overlooked by these prior studies. With ground-truth labels, similarity collapse can be suppressed by explicitly constraining positive and negative pairs to have small and large Hamming distances [2, 3, 27]. This, however, does not fit unsupervised hashing. By imposing similarity distribution prior, our SDC elegantly eliminates the need for training labels.

3 Methodology

To obtain a hash code $b \in \{-1,+1\}^K$ with K bits, we need a hash function h as:

$$\mathbf{b} = h(\mathbf{x}) = \texttt{sign}(\phi(\mathbf{x})), \tag{1}$$

where $\phi : \mathbf{x} \rightarrow \mathbf{f} \in \mathbb{R}^K$ is a (non-)linear mapping function compressing a d-dimensional feature vector $\mathbf{x} \in \mathbb{R}^d$ into a K-dimensional continuous code $\mathbf{f}$. h is learned by optimizing an objective function $\mathcal{L}$. At test time for image retrieval, the Hamming distances between a query code, $\mathbf{b}_p$, and the gallery codes, $\mathbf{b}_q$, of a database can be computed as:

$$\mathcal{D}_h(\mathbf{b}_p, \mathbf{b}_q) = \frac{K}{2}(1 - \cos\theta_{pq}), \tag{2}$$

where $\cos\theta_{pq} = \cos(\mathbf{b}_p, \mathbf{b}_q)$ is the cosine similarity between $\mathbf{b}_p$ and $\mathbf{b}_q$.

It is not differentiable with Eq. (1) in a hashing objective $\mathcal{L}$ due to the non-differentiable `sign` function. A straightforward solution is to remove the `sign` function, whilst minimiz-

ing the quantization error between $\mathbf{f}$ and its hash code $\mathbf{b}$ during training [43] as:

$$\mathcal{L}_q = \frac{1}{N}\sum_{i=1}^{N}(1-\cos\theta_i), \quad \text{and} \quad \cos\theta_i = \cos(\mathbf{f}_i, \mathbf{b}_i), \tag{3}$$

where $\cos\theta_i$ is the cosine similarity between the continuous code $\mathbf{f}_i$ and the hash code counterpart $\mathbf{b}_i = \texttt{sign}(\mathbf{f}_i)$ of i-th sample, and N specifies the training set size. This enables differentiable end-to-end hashing without a straight-through estimator [1, 57] or continuous relaxation [6]. Note, although the continuous codes $\mathbf{f}$ are involved in learning, we describe the learning process directly with hash codes $\mathbf{b}$ hereafter for convenience.

3.1 Hashing by Conventional Similarity Preservation

Prior arts [16, 20, 22, 26, 31, 35, 57] preserve the pairwise similarities of the original continuous feature space during hashing. The loss function is often formulated as:

$$\mathcal{L}_{\text{p}} = \frac{1}{|\mathcal{N}|}\sum_{(i,j)\in\mathcal{N}} |t_{(i,j)} - s_{(i,j)}|^p, \tag{4}$$

where $p = 2$, $t_{(i,j)} = \cos(\mathbf{x}_i, \mathbf{x}_j)$ is the similarity reconstruction target for $\mathbf{x}_i$ and $\mathbf{x}_j$ drawn from a training set $\mathbf{X} \in \mathbb{R}^{N\times d}$, $\mathcal{N}$ is a set of selected sample pairs, $|\mathcal{N}|$ is the set cardinality, and $s_{(i,j)} = \cos(\mathbf{b}_i, \mathbf{b}_j)$ is the similarity of hash codes. Each hash code is obtained with a hash function $\mathbf{b} = h(\mathbf{x})$ (Eq. (1)).

As discussed earlier, similarity preservation based unsupervised hashing methods suffer from a ***similarity collapse*** problem, as indicated by the severe overlapping in the hash code similarity scores of positive and negative pairs (Fig. 1b). Intuitively, this would lead to suboptimal retrieval performance.

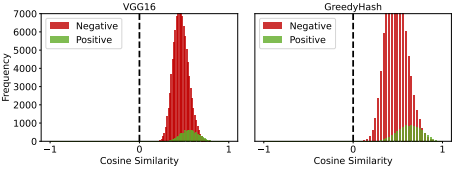


Figure 2: Pairwise similarity distribution of CIFAR10 using 100k randomly chosen negative pairs and 10k positive pairs. **(Left)** VGG16 features. **(Right)** 64-bits hash codes of GreedyHash [57]. For easy explanation, positive and negative labels are included (*i.e.* descriptive purpose only), and were not used during the actual unsupervised training.

As a concrete example, we examine the pairwise similarity distribution of CIFAR10 in the Hamming space. From Fig. 2 we observe that the distribution of hash code similarities is mainly concentrated in the positive region. This is because similarity preservation (*i.e.*, Eq. (4)) would directly inherit the similarity bias of the original feature space (VGG-16 features in this case). As a result, the similarity range of Hamming space is leveraged only at a limited degree, giving rise to the similarity collapse problem.

3.2 Similarity Distribution Calibration

Similarity Distribution Calibration (SDC) is designed particularly for alleviating the similarity collapse problem. The idea is to align the empirical hash code similarity distribution of the training data with a calibration distribution with sufficient spread across the entire similarity range. To measure the discrepancy between two probability distributions for similarity calibration, we adopt the Wasserstein distance with an elegant solution based on *inverse*

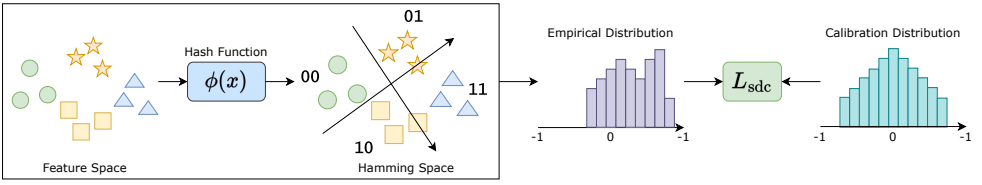


Figure 3: Pipeline of our proposed Similarity Distribution Calibration (SDC). We first map the original features into the Hamming space through a learnable hash function. Next, we construct the empirical hash code similarity distribution. Finally, we minimize the Wasserstein distance (L_{sdc}) between the empirical distribution and a calibration distribution.

Cumulative Distribution Function (iCDF) [49, 51]. Formally, we consider the hash code similarity s as a random variable with the iCDF F conditioned on the *feature similarities* t. Our Wasserstein distance-based calibration is formulated as:

$$\int_0^1 |F(z) - C(z)| dz, \tag{5}$$

where z is the quantile with the interval of $[0,1]$, and C is the iCDF of the calibration distribution. The pipeline of SDC is depicted in Fig. 3.

Approximation. We can estimate F by collecting the pairwise similarities of hash codes and sorting them in the ascending order of t. Concretely, we evenly divide the probability range $[0,1]$ into $|\mathcal{N}|$ bins and then aggregate per-bin calibration as:

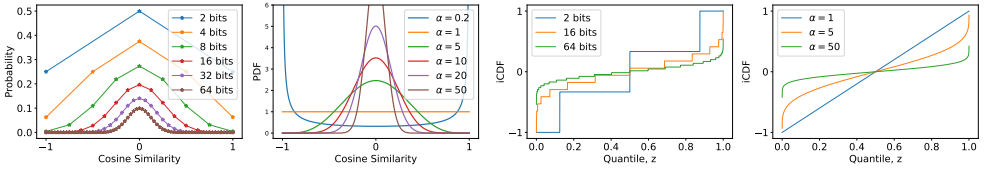
$$\mathcal{L}_{\text{sdc}} = \frac{1}{|\mathcal{N}|} \sum_{s_i \in \mathcal{N}} \left| s_i - C\left(\frac{2i-1}{2|\mathcal{N}|}\right) \right|, \tag{6}$$

where s_i is i-th sorted hash code pairwise similarity. $\mathcal{L}_{\text{sdc}}$ has a similar form to $\mathcal{L}_{\text{p}}$. $\mathcal{L}_{\text{p}}$ with $p=1$ is essentially minimizing the Wasserstein distance between original feature similarity and hash code similarity distribution. Sorting by the order of t conditions the iCDF F, balancing between feature similarity preservation and Hamming similarity range usage.

Instantiation. In general, any distribution with sufficient capacity spread is suited for calibration distribution. As an instantiation, we consider ***beta distribution***, $\texttt{Beta}(\alpha,\beta)$ with α and β the two positive shape parameters. We set $\alpha=\beta$ for simpler symmetric beta distribution. $\texttt{Beta}(\alpha,\beta)$ is chosen as its iCDF is bounded to $[0,1]$, making it easy to be transformed to the target similarity range (*e.g.*, $[-1,1]$ for cosine similarity in our case) when minimizing $\mathcal{L}_{\text{sdc}}$.There is no prior knowledge about the optimal parameter value. However, as illustrated in Fig. 4b, the shapes of probability density functions (PDF) over different parameter values all meet the requirements (*i.e.*, fully utilizing the entire similarity range) as calibration distribution. Empirically, we find that $\alpha=\beta=5$ works generally (approximating a Gaussian distribution of $N(\mu=0,\sigma=0.3)$).

Hash buckets perspective. We interpret the SDC from the hash buckets perspective. It is assumed that an optimal hash function should encode similar items with the same hash code (preserved similarity) and fully utilize the Hamming space (decorrelated and balanced bit) [59]. In an ideal inverted file system [23], K-bits hash codes can form 2^K hash buckets all of which are used at equal size. Thus any two hash codes can be sampled uniformly.

For the balanced bits case, the probability that one bit differs is 0.5, and that d bits differ is $(0.5)^d(0.5)^{K-d}$. There are $\binom{K}{d}$ variants for d bits being different. Thus, the probability



(a) (b) (c) (d)

Figure 4: **(a)** Probability mass function (PMF) of $\mathtt{B}(K,0.5)$ and **(b)** probability density function (PDF) of $\mathtt{Beta}(\alpha,\beta)$ distribution with different α/β values. **(c)** Inverse cumulative density function (iCDF) of $\mathtt{B}(K,0.5)$ and **(d)** $\mathtt{Beta}(\alpha,\beta)$ distribution with different α/β values. Note that we set $\alpha=\beta$ for symmetric PDF.

that the Hamming distance between two uniformly sampled K-bits hash codes equals to d is:

$$\binom{K}{d}(0.5)^d(0.5)^{K-d}=\frac{1}{2^K}\binom{K}{d}. \tag{7}$$

This is equivalent to the probability mass function of a binomial distribution $\mathtt{B}(K,0.5)$. We plot the result in Fig. 4a and the iCDF of Eq. (7) in Fig. 4c. We see that as K varies from 2 to 64, the similarity distribution is similar to the beta distribution with $\alpha=\beta\to\infty$ (see Fig. 4d). This means that for learning an optimal hash function, we should produce hash codes with their pairwise similarity distribution similar to a binomial distribution. The case that Hamming space is not fully used suggests imbalanced hash bucket sizes, meaning a biased similarity distribution and similarity collapse emerges.

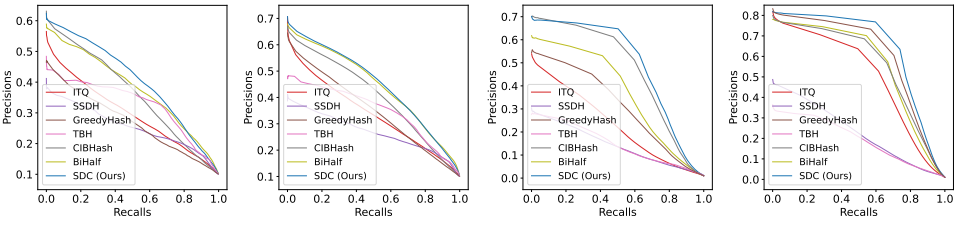
Remarks. The feature similarity scores are used to sort the hash code counterparts while constructing the iCDF. However, unlike the conventional strategy preserving *individual* pairwise similarity scores rigidly, our SDC leverages the distribution of feature similarity scores *holistically*. We observe that pairwise feature similarities could vary over mini-batches. During calibration, we apply sorting to rank them before aligning their corresponding hash code similarities with the prior distribution. As a result, our method does not conduct one-to-one alignment between feature similarity and prior distribution. This property could be understood as a type of stochastic noise during optimization, in the spirit of SGD. Critically, SDC maintains the cluster structure of original features better (Fig. 7), making it more discriminative for image retrieval.

3.3 Overall Learning Objective

For model training, we deploy the overall objective loss function as $\mathcal{L}=\mathcal{L}_{\text{sdc}}+\lambda_q\mathcal{L}_q+\lambda_{\text{cl}}\mathcal{L}_{\text{cl}}$ where $\mathcal{L}_q$ is the quantization loss (Eq. (3)), $\mathcal{L}_{\text{cl}}$ is contrastive learning loss [9] as adopted by recent contrastive hashing methods [48, 65, 66], both λ_{cl} and λ_q are hyper-parameters. We simply set $\lambda_q=1$ and $\lambda_{\text{cl}}=1$ unless mentioned otherwise. See supplementary material for the detailed algorithm.

4 Experiments

Datasets. We consider both coarse category-level and fine-grained instance-level image retrieval tasks in our experiments. Following [14, 35, 48, 54, 57], we use 4 category-level



(a) CIFAR10 (b) ImageNet100

Figure 5: PR curves on CIFAR10 and ImageNet100. For each dataset, the left and right correspond to 16-bits and 64-bits hash codes. Feature: VGG-16.

datasets: i) **CIFAR-10** [30], ii) **NUS-WIDE** [10], and iii) **MS-COCO** [39]. With an ImageNet pre-trained model, we also choose iv) **ImageNet100** (a subset of ImageNet [12] as first used by [6] and later by supervised deep hashing works), which was ignored by previous unsupervised hashing works. For evaluating instance-level retrieval tasks, three popular datasets are chosen including i) **GLDv2** [60], ii) $\mathcal{R}$**Oxf** [46, 50], and iii) $\mathcal{R}$**Paris** [47, 50].

Evaluation metrics. Following previous works [35, 48, 57], we measure the model performance with mean Average Precision (mAP) at top 1000 (mAP@1K) for single-labeled datasets (*i.e.*, ImageNet100 and CIFAR10), while top 5000 (mAP@5K) for multi-labeled datasets (*i.e.*, NUS-WIDE and MS-COCO). Note, for instance-level retrieval tasks, we follow the evaluation protocol of [4, 19] and use mAP@100 as the evaluation metric. For statistical stability, we run 3 trials for each experiment and report the average of per-trial best results on the validation set.

Competitors. We consider 3 classic unsupervised hashing methods [16, 22, 59], and 7 recent state-of-the-art unsupervised deep hashing methods [35, 48, 54, 57, 61, 65, 66].

Implementation details. For fair comparisons, we follow the existing experimental protocol [35, 48, 54, 57]. We use the same pretrained feature extractor (*frozen*) as the competitors (ResNet50 [18] for NSH [66], ViT-B/16 [13] for WCH [65], and VGG-16 [55] for the rest). We test three common code lengths: 16, 32, and 64 bits. We train all the compared methods using Adam [28] optimizer for 100 epochs with a learning rate of 0.0001 and a batch size of 64, with a single exception in TBH [54], for which a batch size of 400 is used and 1000 epochs are required. Note that we re-implemented most competing methods based on the original released codes. Our reimplementation can reproduce the reported performances under the original setting. This allows us to evaluate all the models fairly under a single setting sharing the same datasets, testing protocols, and network architectures. More experimental details including the training/query/gallery splits for each dataset and implementation details including hyperparameters are given in the supplementary material.

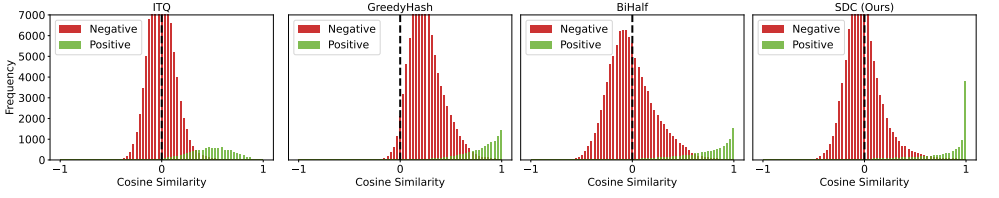
4.1 Comparative Results

Coarse category-level retrieval results. We report the retrieval results of our SDC and prior art alternatives on three datasets in Table 1. We see that SDC outperforms consistently all the competitors, suggesting the importance of similarity collapse as revealed in this work. The precision-recall (PR) curves in Fig. 5 show that our SDC (blue curves) excels across different recall rates, especially at low bit cases (*i.e.*, 16-bits).

Instance-level retrieval results. We also evaluate our model on instance-level image re-

Methods	Reference	CIFAR10			ImageNet100			NUSWIDE			MS-COCO		
		16	32	64	16	32	64	16	32	64	16	32	64
VGG16													
LsH [42]	STOC'98	23.9	29.6	37.6	14.7	29.7	48.7	51.0	59.3	67.1	45.2	51.6	59.8
SH [59]	NeurIPS'08	41.8	42.1	43.5	35.1	50.9	60.9	63.0	60.9	64.0	59.4	64.8	66.2
ITQ [16]	TPAMI'12	46.8	51.3	54.4	45.5	62.1	72.7	73.2	75.0	77.1	67.6	72.9	75.4
SSDH [53]	IJCAI'18	41.0	39.6	38.5	32.3	40.1	44.6	66.8	67.8	66.7	53.9	56.7	57.4
GreedyHash [57]	NeurIPS'18	44.9	51.9	55.7	54.4	68.7	74.7	70.0	76.2	79.3	66.8	73.2	77.4
TBH [52]	CVPR'20	48.2	50.2	50.7	42.9	44.5	48.3	75.8	77.8	78.5	68.8	72.6	74.8
CIBHash† [43]	IJCAI'21	56.2	59.2	61.2	63.9	71.4	74.6	77.1	79.7	80.9	73.3	77.0	78.5
BiHalf [65]	AAAI'21	54.7	58.1	60.6	60.7	71.2	76.0	77.4	80.1	81.9	71.2	75.6	78.0
SDC†	**Ours**	**59.8**	**64.0**	**66.3**	**72.8**	**78.5**	**80.6**	**80.7**	**82.3**	**83.4**	**76.9**	**79.8**	**81.2**
ResNet50													
NSH† [27]	IJCAI'22	70.6*	73.3*	75.6*	-	-	-	75.8*	81.1*	82.4*	74.6*	77.4*	78.3*
SDC†	**Ours**	**74.2**	**75.8**	**78.4**	**80.7**	**83.8**	**85.7**	**81.2**	**83.2**	**84.2**	**78.3**	**81.1**	**82.6**
ViT-B/16													
WCH† [63]	ACCV'22	77.5	79.3	80.6	69.4	76.9	80.8	70.7	75.6	78.6	73.0	78.8	81.4
SDC†	**Ours**	**87.4**	**88.4**	**89.0**	**76.4**	**82.6**	**84.9**	**81.8**	**83.3**	**84.0**	**79.2**	**83.3**	**84.5**

Table 1: Unsupervised hashing results. *: Originally reported. †: Using contrastive learning.



(a) 37.92% (b) 37.33% (c) 35.97% (d) **28.18%**

Figure 6: Analysis of the similarity collapse problem on ImageNet100. We plot the hamming distance histograms for 10000 positive and 100000 negative random pairs with 64-bits hash codes. For similarity collapse quantification, we use the intersection between the two histograms as the metric, lower is better. Note that, the positive and negative labels are used for illustration purpose only, but not used during model unsupervised training.

trieval tasks. For training efficiency, we turn off the contrastive loss (*i.e.*, $\lambda_{\text{cl}} = 0$). We follow the evaluation protocol of [19][1]. We use three datasets, namely GLDv2 [60], $\mathcal{R}$Oxf, and $\mathcal{R}$Paris [50]. Note, due to no training data with $\mathcal{R}$Oxf and $\mathcal{R}$Paris, we use the training set of GLDv2 for model training for all datasets. As shown in Table 2, our SDC still outperforms consistently the state-of-the-art similarity preservation based methods (*i.e.*, GreedyHash [57], and BiHalf [65]) by a large margin. This indicates that the superiority of our SDC generalizes from coarse category retrieval to fine-grained instance retrieval, even in the presence of a distributional shift between the training and test sets.

Methods	GLDv2		$\mathcal{R}$Oxf		$\mathcal{R}$Paris	
	128	512	128	512	128	512
ITQ [16]	5.2	11.3	1.6	5.4	4.8	12.3
GreedyHash [57]	3.8	7.9	15.8	34.2	34.9	52.8
BiHalf [65]	4.0	6.7	20.2	33.3	42.0	52.0
SDC (Ours)	**6.3**	**12.1**	**27.1**	**40.8**	**50.3**	**63.8**
Original features	13.8		51.0		71.5	

Table 2: Instance-level image retrieval results of representative unsupervised hashing methods on GLDv2, $\mathcal{R}$Oxf-**Hard** and $\mathcal{R}$Paris-**Hard**. Original features: 2048D R50-DELG features [4] with the cosine similarity.

4.2 Further Analysis

Similarity collapse analysis. We first examine the similarity collapse problem. We study three representative similarity preservation-based hashing methods (ITQ, GreedyHash, Bi-

[1] Please see supplementary material for implementation details.

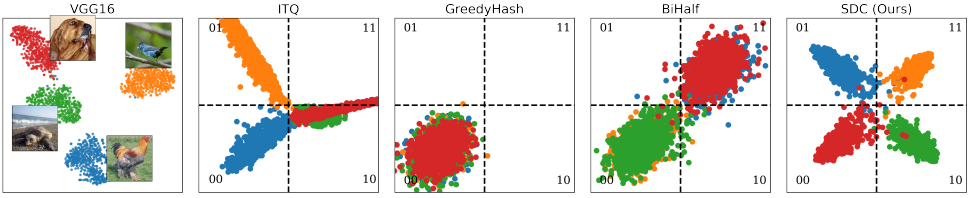


Figure 7: **(a)** The t-SNE visualization of VGG16 features of 4 object classes of ImageNet100. **(b-e)** Continuous 2-bit codes before quantization derived by different unsupervised hashing methods. Dotted lines denote the separation of Hamming space.

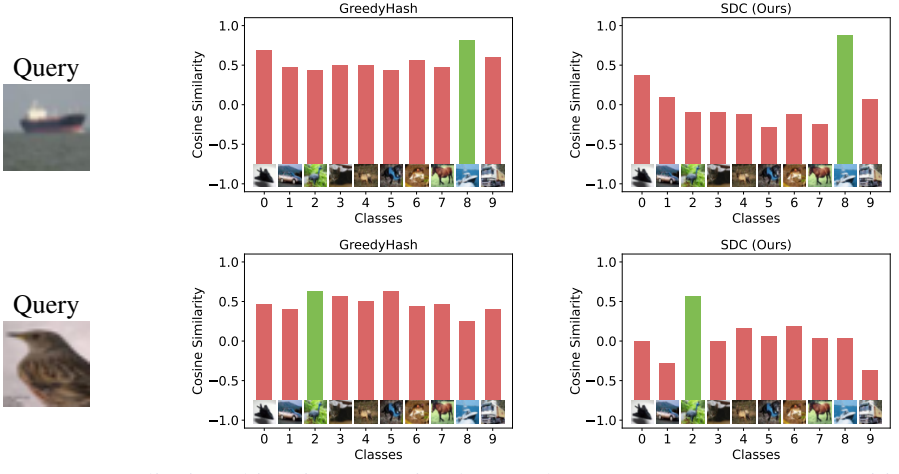


Figure 8: Two qualitative object image retrieval examples on CIFAR10. Green: Positive class; Red: Negative class.

half) in comparison with our SDC. To quantify this collapse, we compute the intersection between the cosine similarity histogram of positive and negative pairs. Higher intersection rates suggest worse collapses with lower discriminating ability. We use 64-bits hash codes. To compute the two histograms, we sample 10k positive and 100k negative random pairs of ImageNet100. Fig. 6 presents the degree of similarity collapse in the order of ITQ $>$ GreedyHash $>$ BiHalf $>$ SDC. This verifies again that our method is effective in alleviating this collapse problem.

Hash code visualization. We visualize continuous hash codes in a proof-of-concept setting. Specifically, we examine the behaviour of ITQ, GreedyHash, BiHalf and SDC in learning a 2-bits hash function over 4 object classes (`cock`, `indigo-bunting`, `loggerhead`, `bloodhound`) from ImageNet100. We use the VGG-16 features. We observe from Fig. 7 that whilst the original features are already well separable, different methods behave differently. For example, simply preserving the original similarity, GreedyHash collapses the similarity scores completely across all the classes. With a code balance layer on top, BiHalf partly reduces the degree of collapsing to two groups. Through aligning the similarity distribution with a calibration distribution, our SDC solves this collapsing problem well, even further separating the originally confusing two classes (`cock` and `bloodhound`). This validates the unexceptional potential of SDC in improving the retrieval performance over

previous methods.

Qualitative evaluation. For visual analysis, we provide a couple of image retrieval examples on CIFAR10. It is evident in Fig. 8 that our SDC can identify the positive class more confidently with a more distinctive separation between positive and negative classes compared to GreedyHash. This indicates the superior discrimination ability of our similarity distribution calibration idea in unsupervised hashing.

5 Conclusion

We have presented a simple yet effective *Similarity Distribution Calibration* (SDC) method for unsupervised hashing. This is particularly designed to alleviate the largely ignored *similarity collapse* problem suffered by the existing similarity preservation-based unsupervised hashing methods. Concretely, we minimize the Wasserstein distance between the distribution of Hamming similarities and a calibration distribution with a sufficient capacity range. As a result, the low similarity range of hash code can be better exploited for improved discriminating ability. Extensive experiments on both coarse and fine-grained image retrieval tasks validated the advantage of our method over the state-of-the-art alternatives.

References

[1] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[2] Fatih Cakir, Kun He, Sarah A. Bargal, and Stan Sclaroff. Mihash: Online hashing with mutual information. In *International Conference on Computer Vision*, 2017.

[3] Fatih Cakir, Kun He, Sarah Adel Bargal, and Stan Sclaroff. Hashing with mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[4] Bingyi Cao, André Araujo, and Jack Sim. Unifying deep local and global features for image search. In *European Conference on Computer Vision*, 2020.

[5] Yue Cao, Bin Liu, Mingsheng Long, and Jianmin Wang. Hashgan: Deep learning to hash with pair conditional wasserstein gan. In *Computer Vision and Pattern Recognition*, 2018.

[6] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Philip S. Yu. Hashnet: Deep learning to hash by continuation. In *International Conference on Computer Vision*, 2017.

[7] Miguel A Carreira-Perpinán and Ramin Raziperchikolaei. Hashing with binary autoencoders. In *Computer Vision and Pattern Recognition*, 2015.

[8] Junjie Chen, William K Cheung, and Anran Wang. Learning deep unsupervised binary codes for image retrieval. In *International Joint Conference on Artificial Intelligence*, 2018.

[9] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, 2020.

[10] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yan-Tao Zheng. Nus-wide: A real-world web image database from national university of singapore. In *Conference on Image and Video Retrieval*, 2009.

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

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition*, 2009.

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[14] Lixin Fan, Kam Woh Ng, Ce Ju, Tianyu Zhang, and Chee Seng Chan. Deep polarized network for supervised learning of accurate binary hashing codes. In *International Joint Conference on Artificial Intelligence*, 2020.

[15] Yunchao Gong, Sanjiv Kumar, Vishal Verma, and Svetlana Lazebnik. Angular quantization-based binary codes for fast similarity search. In *Advances in Neural Information Processing Systems*, 2012.

[16] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012.

[17] Yifan Gu, Haofeng Zhang, Zheng Zhang, and Qiaolin Ye. Unsupervised deep triplet hashing with pseudo triplets for scalable image retrieval. *Multimedia Tools and Applications*, 2019.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition*, 2016.

[19] Jiun Tian Hoe, Kam Woh Ng, Tianyu Zhang, Chee Seng Chan, Yi-Zhe Song, and Tao Xiang. One loss for all: Deep hashing with a single cosine similarity based learning objective. In *Advances in Neural Information Processing Systems*, 2021.

[20] Mengqiu Hu, Yang Yang, Fumin Shen, Ning Xie, and Heng Tao Shen. Hashing with angular reconstructive embeddings. *IEEE Transactions on Image Processing*, 2018.

[21] Shanshan Huang, Yichao Xiong, Ya Zhang, and Jia Wang. Unsupervised triplet hashing for fast image retrieval. In *Thematic Workshops of ACM Multimedia*, 2017.

[22] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Annual ACM Symposium on Theory of Computing*, 1998.

[23] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010.

[24] Sheng Jin, Hongxun Yao, Xiaoshuai Sun, and Shangchen Zhou. Unsupervised semantic deep hashing. *Neurocomputing*, 2019.

[25] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 2021.

[26] Ian Jolliffe. Principal component analysis and factor analysis. *Principal Component Analysis*, 1986.

[27] Mete Kemertas, Leila Pishdad, Konstantinos G Derpanis, and Afsaneh Fazly. Rankmi: A mutual information maximizing ranking loss. In *Computer Vision and Pattern Recognition*, 2020.

[28] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[29] Weihao Kong and Wu-jun Li. Isotropic hashing. In *Advances in Neural Information Processing Systems*, 2012.

[30] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[31] Brian Kulis and Trevor Darrell. Learning to hash with binary reconstructive embeddings. In *Advances in Neural Information Processing Systems*, 2009.

[32] Shuyan Li, Zhixiang Chen, Jiwen Lu, Xiu Li, and Jie Zhou. Neighborhood preserving hashing for scalable video retrieval. In *International Conference on Computer Vision*, 2019.

[33] Wu-Jun Li, Sheng Wang, and Wang-Cheng Kang. Feature learning based deep supervised hashing with pairwise labels. In *International Joint Conference on Artificial Intelligence*, 2016.

[34] Yang Li, Yapeng Wang, Zhuang Miao, Jiabao Wang, and Rui Zhang. Contrastive self-supervised hashing with dual pseudo agreement. *IEEE Access*, 2020.

[35] Yunqiang Li and Jan van Gemert. Deep unsupervised image hashing by maximizing bit entropy. In *AAAI Conference on Artificial Intelligence*, 2021.

[36] Kevin Lin, Jiwen Lu, Chu-Song Chen, and Jie Zhou. Learning compact binary descriptors with unsupervised deep neural networks. In *Computer Vision and Pattern Recognition*, 2016.

[37] Mingbao Lin, Rongrong Ji, Hong Liu, and Yongjian Wu. Supervised online hashing via hadamard codebook learning. In *ACM International Conference on Multimedia*, 2018.

[38] Mingbao Lin, Rongrong Ji, Hong Liu, Xiaoshuai Sun, Shen Chen, and Qi Tian. Hadamard matrix guided online hashing. *International Journal of Computer Vision*, 2020.

[39] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, 2014.

[40] Bin Liu, Yue Cao, Mingsheng Long, Jianmin Wang, and Jingdong Wang. Deep triplet quantization. In *ACM International Conference on Multimedia*, 2018.

[41] Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. Supervised hashing with kernels. In *Computer Vision and Pattern Recognition*, 2012.

[42] Wei Liu, Cun Mu, Sanjiv Kumar, and Shih-Fu Chang. Discrete graph hashing. In *Advances in Neural Information Processing Systems*, 2014.

[43] Xiao Luo, Daqing Wu, Chong Chen, Minghua Deng, Jianqiang Huang, and Xian-Sheng Hua. A survey on deep hashing methods. *arXiv preprint arXiv:2003.03369*, 2020.

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

[45] Mohammad Norouzi, David J Fleet, and Russ R Salakhutdinov. Hamming distance metric learning. In *Advances in Neural Information Processing Systems*, 2012.

[46] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Computer Vision and Pattern Recognition*, 2007.

[47] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *Computer Vision and Pattern Recognition*, 2008.

[48] Zexuan Qiu, Qinliang Su, Zijing Ou, Jianxing Yu, and Changyou Chen. Unsupervised hashing with contrastive information bottleneck. In *International Joint Conference on Artificial Intelligence*, 2021.

[49] Julien Rabin, Gabriel Peyré, Julie Delon, and Marc Bernot. Wasserstein barycenter and its application to texture mixing. In *International Conference on Scale Space and Variational Methods in Computer Vision*, 2011.

[50] Filip Radenovic, Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. Revisiting oxford and paris: Large-scale image retrieval benchmarking. In *Computer Vision and Pattern Recognition*, 2018.

[51] Aaditya Ramdas, Nicolás García Trillos, and Marco Cuturi. On wasserstein two-sample testing and related families of nonparametric tests. *Entropy*, 2017.

[52] Fumin Shen, Yan Xu, Li Liu, Yang Yang, Zi Huang, and Heng Tao Shen. Unsupervised deep hashing with similarity-adaptive and discrete optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[53] Yuming Shen, Li Liu, and Ling Shao. Unsupervised binary representation learning with deep variational networks. *International Journal of Computer Vision*, 2019.

[54] Yuming Shen, Jie Qin, Jiaxin Chen, Mengyang Yu, Li Liu, Fan Zhu, Fumin Shen, and Ling Shao. Auto-encoding twin-bottleneck hashing. In *Computer Vision and Pattern Recognition*, 2020.

[55] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[56] Jingkuan Song, Tao He, Lianli Gao, Xing Xu, Alan Hanjalic, and Heng Tao Shen. Binary generative adversarial networks for image retrieval. In *AAAI Conference on Artificial Intelligence*, 2018.

[57] Shupeng Su, Chao Zhang, Kai Han, and Yonghong Tian. Greedy hash: Towards fast optimization for accurate hash coding in cnn. In *Advances in Neural Information Processing Systems*, 2018.

[58] Xiaofang Wang, Yi Shi, and Kris M Kitani. Deep supervised hashing with triplet labels. In *Asian Conference on Computer Vision*, 2016.

[59] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In *Advances in Neural Information Processing Systems*, 2009.

[60] Tobias Weyand, Andre Araujo, Bingyi Cao, and Jack Sim. Google landmarks dataset v2-a large-scale benchmark for instance-level recognition and retrieval. In *Computer Vision and Pattern Recognition*, 2020.

[61] Erkun Yang, Cheng Deng, Tongliang Liu, Wei Liu, and Dacheng Tao. Semantic structure-based unsupervised deep hashing. In *International Joint Conference on Artificial Intelligence*, 2018.

[62] Erkun Yang, Tongliang Liu, Cheng Deng, Wei Liu, and Dacheng Tao. Distillhash: Unsupervised deep hashing by distilling data pairs. In *Computer Vision and Pattern Recognition*, 2019.

[63] Huei-Fang Yang, Kevin Lin, and Chu-Song Chen. Supervised learning of semantics-preserving hash via deep convolutional neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[64] Zhan Yang, Osolo Ian Raymond, Wuqing Sun, and Jun Long. Deep attention-guided hashing. *IEEE Access*, 2019.

[65] Jiaguo Yu, Huming Qiu, Dubing Chen, and Haofeng Zhang. Weighted contrastive hashing. In *Asian Conference on Computer Vision*, 2022.

[66] Jiaguo Yu, Yuming Shen, Menghan Wang, Haofeng Zhang, and Philip HS Torr. Learning to hash naturally sorts. In *International Joint Conference on Artificial Intelligence*, 2022.

[67] Jiaguo Yu, Yuming Shen, Haofeng Zhang, Philip H. S. Torr, and Menghan Wang. Learning to hash naturally sorts. In *International Joint Conference on Artificial Intelligence*, 2022.

[68] Shuying Yu, Xian-Ling Mao, Wei Wei, and Heyan Huang. Unsupervised deep hashing via adaptive clustering. In *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data*, 2021.

[69] Li Yuan, Tao Wang, Xiaopeng Zhang, Francis EH Tay, Zequn Jie, Wei Liu, and Jiashi Feng. Central similarity quantization for efficient image and video retrieval. In *Computer Vision and Pattern Recognition*, 2020.

[70] Wanqian Zhang, Dayan Wu, Yu Zhou, Bo Li, Weiping Wang, and Dan Meng. Deep unsupervised hybrid-similarity hadamard hashing. In *ACM International Conference on Multimedia*, 2020.

[71] Kang Zhao, Hongtao Lu, and Jincheng Mei. Locality preserving hashing. In *AAAI Conference on Artificial Intelligence*, 2014.

[72] Yuxuan Zhu, Yali Li, and Shengjin Wang. Unsupervised deep hashing with adaptive feature learning for image retrieval. *IEEE Signal Processing Letters*, 2019.

[73] Maciej Zieba, Piotr Semberecki, Tarek El-Gaaly, and Tomasz Trzcinski. Bingan: Learning compact binary descriptors with a regularized gan. In *Proceedings of Advances in Neural Information Processing Systems*, 2018.