Deep Hash Distillation for Image Retrieval

Young Kyun Jang[1], Geonmo Gu[2], Byungsoo Ko[2], Isaac Kang[1], and Nam Ik Cho[1,3]

[1] ECE & INMC, Seoul National University, Korea
[2] NAVER Vision
[3] IPAI, Seoul National University, Korea
{kyun0914, korgm403, kobiso62}@gmail.com, {isaackang, nicho}@snu.ac.kr

Abstract. In hash-based image retrieval systems, degraded or transformed inputs usually generate different codes from the original, deteriorating the retrieval accuracy. To mitigate this issue, data augmentation can be applied during training. However, even if augmented samples of an image are similar in real feature space, the quantization can scatter them far away in Hamming space. This results in representation discrepancies that can impede training and degrade performance. In this work, we propose a novel self-distilled hashing scheme to minimize the discrepancy while exploiting the potential of augmented data. By transferring the hash knowledge of the weakly-transformed samples to the strong ones, we make the hash code insensitive to various transformations. We also introduce hash proxy-based similarity learning and binary cross entropy-based quantization loss to provide fine quality hash codes. Ultimately, we construct a deep hashing framework that not only improves the existing deep hashing approaches, but also achieves the state-of-the-art retrieval results. Extensive experiments are conducted and confirm the effectiveness of our work. Code is at https://github.com/youngkyunJang/Deep-Hash-Distillation

Keywords: Large-scale Image Retrieval, Learning to Hash, Self-distillation

1 Introduction

Especially for retrieval from large-scale databases, *hashing* is essential due to its practicality, *i.e.,* high search speed and low storage cost. By converting high-dimensional data points into compact binary codes with a hash function, the retrieval system can utilize a simple bit-wise XOR operation to define a distance between the images. A wide variety of works have been studied for learning to hash [41,15,43,33,38], and are still being actively pursued to build fast and accurate retrieval systems.

Recently, techniques for hash learning have been significantly advanced by deep learning, which is called *deep hashing*, and its corresponding works are in the spotlight [1,20,11,42,29,39,50,46]. By integrating the hash function into the deep learning framework, the image encoder and hash function are simultaneously learned to generate image hash codes. Regarding the training of deep hashing, the leading techniques are pairwise similarity approaches that use sets of similar or dissimilar image pairs [44,51,3,4,21,22], and global similarity in company with classification approaches that use class labels assigned to images [23,24,47].

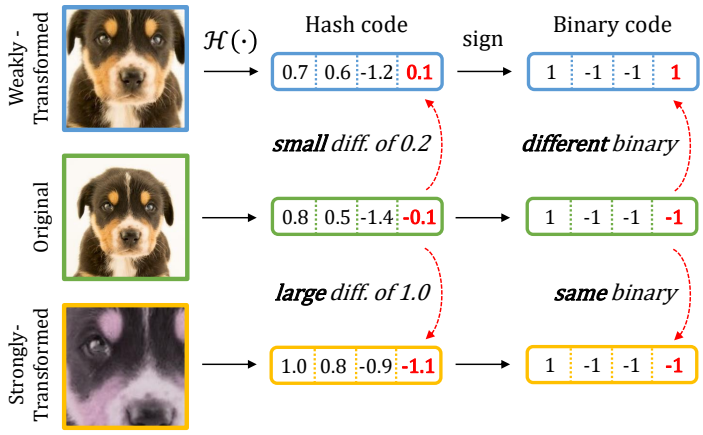


Fig. 1: Visualization of possible problems in deep hashing due to transformations. The continuous hash code generated by a deep hashing model $\mathcal{H}(\cdot)$ is changed when the input is transformed. In consequence, the binary code quantized with sign operation can also be shifted. However, the degree of transformation cannot be properly reflected in the quantized representation.

Since hash-based retrieval systems compute the distance between images with binary codes, corresponding codes need to be quantized with *sign* operation, from the continuous real space to the discrete Hamming space of $\{-1, 1\}$. In this process, the continuously optimized image representation is altered, and quantization error occurs, which in turn degrades the discriminative capability of the hash code. This becomes even more problematic when an input image is transformed and deviated from the original distribution.

To avoid performance degradation due to transformations, the most common solution is to generalize the deep model by training it with augmented data having various transformations. However, it is challenging to apply this augmentation strategy to deep hash training since discrepancy in the representation may occur. Figure 1 shows an example case that may appear: 1) *The sign of the hash code can be shifted with a slight change*. Specifically, the last element of the weakly-transformed image's hash code differs by 0.2 ($-0.1 \rightarrow 0.1$) from the original, but it results in $-1 \rightarrow 1$ shift in the Hamming space. 2) *The sign of the quantized hash code does not shift even with the big change in the hash code*. The last element of a strongly transformed image's hash code differs by 1.0 ($-0.1 \rightarrow -1.1$) from the original, resulting in no shifts in the Hamming space. Namely, the use of strong augmentation in deep hashing increases the discrepancy between Hamming and real space, which hinders finding the optimal binary code.

To resolve this issue, we introduce a novel concept dubbed *Self-distilled Hashing*, which customizes self-distillation [49,40,45,5,32] to prevent severe discrepancy in deep hash training. Specifically, based on the understanding of the relation between cosine distance and Hamming agreement [33,48,19], we minimize the cosine distance between the hash codes of two different views (transformed results) of an image to maximize the Hamming agreement between their binary outcomes. Further for stable learning, we

separate the difficulties of transformations as easy and difficult, and transfer the hash knowledge from easy to difficult, inspired by [8,40,5].

Moreover, we propose two additional training objectives that optimize hash codes to enhance the self-distilled hashing: 1) a hash proxy-based similarity learning, and 2) a binary cross entropy-based quantization loss. The first term allows the deep hashing model to learn global (inter-class) discriminative hash codes with temperature-scaled cosine similarity. The second term contributes to making the hash code naturally move away from the binary threshold in a classification manner with likelihood estimators.

By combining all of our proposals, we construct a **D**eep **H**ash **D**istillation framework (DHD), which yields discriminative and transformation resilient hash codes for fast image retrieval. We conduct extensive experiments on single and multi-labeled benchmark datasets for image hashing evaluation. In addition, we validate the effectiveness of self-distilled hashing using data augmentation on the existing methods [4,3,14,47] and show the performance improvements. Furthermore, we establish that DHD is applicable with a variety of deep backbone architectures including vision Transformers [12,40,34]. Experimental results verify that self-distilled hashing strategy improves the existing works, and entire DHD framework shows the best retrieval performance.

We can summarize our contributions as follows:

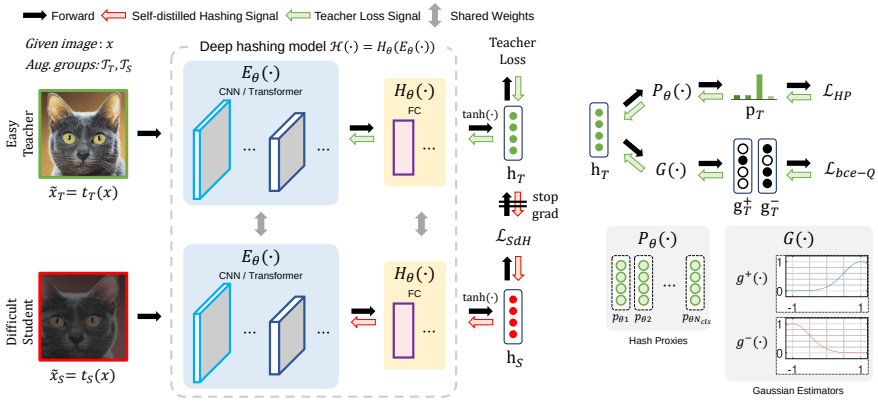
- To the best of our knowledge, this is the first work to address the discrepancy between real and Hamming space provoked by data augmentation in deep hashing.
- With the introduction of self-distilled hashing scheme and training loss functions, we successfully embed the power of augmentations into the hash codes.
- Extensive experiments demonstrate the benefits of our work, improving previous deep hashing methods and achieving the state-of-the-art performances.

2 Related Works

For a better understanding, we present a brief introduction to the deep hashing methods and the research that inspired our proposal. Refer to a survey [41] to see details of the early works in non-deep hashing approaches (ITQ [15], SH [43], KSH [33], SDH [38]).

Deep hashing methods. Hashing algorithms using deep learning techniques such as Convolutional Neural Network (CNN) are leading the mainstream with striking results. For example, CNNH [44] utilizes a CNN to generate compact hash codes by training a network with given pairwise label information. DHN [51] learns hash codes by approximating discrete values with relaxation and trains them with supervised signals. HashNet [4] adopts the inner product to measure pairwise similarity between hash codes and tackles the data imbalance problem by employing weighted maximum likelihood estimation. DCH [3] employs Cauchy distribution to minimize the Hamming distance of the images with the same class label.

Hash center-based methods. There have been several methods to find out class-wise hash representatives (centers), which can provide global similarity to hash codes by including the process of predicting image class labels with hash codes during training [23,24,21,47]. CSQ [47] uses pre-defined orthogonal binary hash targets to guarantee a certain Hamming distance between classes and makes hash codes follow the targets.



(a) Self-distilled hashing. (b) Detailed Teacher loss.

Fig. 2: The overall training process of Deep Hash Distillation (DHD) framework. (a) From two different augmentation groups, namely Teacher $\mathcal{T}_T$ and Student $\mathcal{T}_S$, randomly sampled transformations ($t_T \sim \mathcal{T}_T$, $t_S \sim \mathcal{T}_S$) are individually applied on the input image x to produce $\tilde{x}_T$ and $\tilde{x}_S$. The deep hashing model $\mathcal{H}(\cdot)$ constructed with the deep encoder $E_\theta(\cdot)$ and the hash function $H_\theta(\cdot)$ of Fully-Connected (FC) layers yields two hash codes h_T and h_S which are learned with $\mathcal{L}_{SdH}$. We apply stop gradient operation on h_T for stable training. (b) Additionally, we employ trainable hash proxies $P_\theta(\cdot)$ which are used to calculate the class-wise prediction p_T with h_T to optimize with $\mathcal{L}_{HP}$, and pre-defined Gaussian estimator $G(\cdot)$ to regularize h_T with $\mathcal{L}_{bce\text{-}Q}$.

DPN [14] employs randomly assigned target vectors with maximal inter-class similarity and utilizes bit-wise hinge-like loss. Unlike DPN and CSQ, which use a hash target that is not trainable, in our DHD, the hash center is set as a trainable proxy which jointly learns the similarity with the hash codes during training.

Self-distillation. Inspired by knowledge distillation [18], self-distillation emerged as a concept that employs a single network to generalize itself in a self-taught fashion, and plenty of works demonstrated its benefits in improving deep model performance [49,40,45,5]. Many of them utilize a simple Siamese architecture [8] to explore and learn the visual representation with data augmentation, by contrasting two different augmented results of one image. Similarly, we conduct the self-distillation with augmentations in deep hashing to see the hash codes of two different views of an image simultaneously. Additionally, in accordance with the characteristics of hashing, we consider a method of minimizing the cosine distance that behaves similarly to the distance in the Hamming space to reduce the representation discrepancy during model training.

3 Method

The goal of a deep hashing model $\mathcal{H}$ of Deep Hash Distillation (DHD) is to map an input image x to a K-dimensional binary code $\mathrm{b} \in \{-1, 1\}^K$ in Hamming space. For this purpose, $\mathcal{H}$ is optimized to find a high quality real-valued hash code h, and then

sign operation is utilized to quantize h as b. Instead of including non-differentiable quantization process in model training, we learn $\mathcal{H}$ in the real space to estimate optimal b with continuously relaxed h while fully exploiting the power of data augmentation. We notate trainable components with θ as a subscript. In the following, h becomes robust to transformations in 3.1, and becomes discriminative and binary-like in 3.2.

3.1 Self-distilled Hashing

In general, $\mathcal{H}$ is trained in the real space to obtain discriminative h, which should maintain its property in the Hamming space even if quantized to b. Therefore, it is important to align h and b to carry a similar representation during training. However, when data augmentation is applied to the training input and the following change occurs in h, there can be misalignment between h and b, as shown in Figure. 1. Thus, direct use of augmentations can cause discrepancies in the representation between h and b, which degrades retrieval performance as we observed in Sec 4.3.

Hamming distance as cosine distance. It is noteworthy that the Hamming distance between the binary codes can be interpreted as cosine distance (1-cosine similarity)[4] [33,48,19]. That is, the cosine similarity between hash codes h_i and h_j can be utilized to approximate the Hamming distance between the binary codes b_i and b_j as:

$$\mathcal{D}_{\mathrm{H}}(\mathrm{b}_i, \mathrm{b}_j) \simeq \frac{K}{2}(1 - \mathcal{S}\left(\mathrm{h}_i, \mathrm{h}_j\right)) \tag{1}$$

where $\mathrm{b}_i = \mathrm{sign}(\mathrm{h}_i)$, $\mathrm{b}_j = \mathrm{sign}(\mathrm{h}_j)$, $\mathcal{D}_{\mathrm{H}}(\cdot,\cdot)$ denotes Hamming distance, $\mathcal{S}(\cdot,\cdot)$ denotes cosine similarity. That is, the minimized cosine distance between the hash codes minimizes the Hamming distance between the binary codes.

Easy-teacher and difficult-student. As shown in Figure 2a, we propose a self-distilled hashing scheme, which supports the training of deep hashing models with augmentations. We employ weight-sharing Siamese structure [26] to contrast hash codes of two different views (augmentation results) of an image at once. According to the observations in self-distillation works [8,5], keeping the output representation of one branch steady has a significant impact on performance gain. Therefore, we configure two separate augmentation groups to provide input views with different difficulties of transformation: one is weakly-transformed easy teacher $\mathcal{T}_T$, and the other is strongly-transformed difficult student $\mathcal{T}_S$. Here, we control the difficulty in a stochastic sampling manner as: employing the same hyper-parameter s_T to all transformations in the group, and make them occur less (weakly) or more (strongly) by scaling their own probability of occurrence. While this manner makes the teacher representation stable, it has the advantage that few extreme examples that produce unstable results are not completely ruled out and contribute to learning. Besides, we stop the gradient of the teacher view's corresponding hash codes to avoid collapsing into trivial solutions [8,5].

Loss computation. For a given image x, self-distillation is conducted with image views as: $\tilde{x}_T = t_T(x)$ and $\tilde{x}_S = t_S(x)$, where t_T, t_S are randomly sampled transformations

[4] Refer supplementary material for proof.

from $\mathcal{T}_T, \mathcal{T}_S$, respectively. The deep encoder E_θ and the hash function H_θ take $\tilde{x}_T$ and $\tilde{x}_S$ as inputs and produce corresponding hash code h_T and h_S. Then, the proposed Self-distilled Hashing (SdH) loss is computed as:

$$\mathcal{L}_{SdH}(\mathrm{h}_T, \mathrm{h}_S) = 1 - \mathcal{S}(\mathrm{h}_T, \mathrm{h}_S) \tag{2}$$

Optimizing $\mathcal{H}$ with $\mathcal{L}_{SdH}$ results in the alignment of h_T and h_S, and thus b_T and b_S as follows Eqn. 1, which in turn reduces the discrepancy in representation between two differently transformed output binary codes.[5]

Flexibility. Note that self-distilled hashing is applicable to the other common deep hashing models [4,3,14,47] with regard to exploiting data augmentation during training, as shown in Section 4.3. Furthermore, various backbones [27,17,12,40,34] can be utilized as deep encoder, and any hash function H_θ configuration is compatible. For simplicity, we employ a single FC layer to obtain a hash code of the desired bits, and apply *tanh* operation at the end to be bound in $[-1, 1]$.

3.2 For Better Teacher

Besides self-distilled hashing, additional training signals such as supervised learning loss, and quantization loss are required to obtain the discriminative hash codes. We only employ teacher hash codes to compute the losses, in order to transfer the learned hash knowledge to the student's codes.

Proxy-based similarity learning. Supervised hash similarity learning with pre-defined orthogonal binary hash targets has shown great performance [47,14,19]. However, the hash target has limitations in that 1) it requires a complex initialization process, and 2) it allocates the same Hamming distance between centers so detailed distances according to semantic similarity cannot be learned. Therefore, as shown in Figure 2b, we introduce a proxy-based representation learning [36,7,16] in deep hashing by using a collection of trainable hash proxies P_θ. It has the advantage that the proxies are simply initialized with randomness, and being able to learn semantic similarity into the proxies. In terms of training, we first use P_θ to compute class-wise prediction p_T with h_T as:

$$\mathrm{p}_T = [\mathcal{S}(p_{\theta 1}, \mathrm{h}_T), \mathcal{S}(p_{\theta 2}, \mathrm{h}_T), ..., \mathcal{S}(p_{\theta N_{cls}}, \mathrm{h}_T)] \tag{3}$$

where p_{θ_i} is a hash proxy assigned to each of the i-th class and N_{cls} denotes the number of classes to be distinguished. Then, we use p_T to learn the similarity with class label y by computing Hash Proxy (HP) loss as:

$$\mathcal{L}_{HP}(\mathrm{y}, \mathrm{p}_T, \tau) = H\left(\mathrm{y}, \mathrm{Softmax}(\mathrm{p}_T/\tau)\right) \tag{4}$$

where τ is a temperature scaling hyper-parameter, $H(\mathrm{u}, \mathrm{v}) = -\sum_k u_k \log v_k$ is a cross entropy, and Softmax operation is applied along the dimension of p_T. Note that, similar to Eqn 1, $\mathcal{L}_{HP}$ is designed to learn Hamming agreement with temperature scaling.

[5] We provide a pseudo-code implementation in supplementary material.

Reducing quantization error. To make continuous hash code elements act like binary bits, the deep hashing methods [3,4,23,47] aim to reduce the quantization error by minimizing the distance (e.g. Euclidean) between the hash code bit and its closest binary goal ($+1$ or -1) in a regression manner. However, since the purpose of hashing is to classify the sign of each bit, it is a more natural choice to view it as a binary classification: maximum likelihood problem. Hence, we adopt a pre-defined Gaussian distribution estimator $g(h)$ of mean m and standard deviation σ as:

$$g(h) = \exp\left(-\frac{(h-m)^2}{2\sigma^2}\right) \tag{5}$$

to evaluate the binary likelihood of hash code element h. By employing two estimators: g^+ of $m = 1$, and g^- of $m = -1$ with the same σ, we compute the likelihoods and a Binary Cross Entropy-based (BCE) quantization loss as:

$$\mathcal{L}_{bce\text{-}Q}(\mathrm{h}_T) = \frac{1}{K}\sum_{k=1}^{K}\left(H_b\left(b_k^+, g_k^+\right) + H_b\left(b_k^-, g_k^-\right)\right) \tag{6}$$

where $H_b(u,v) = -\left(u\log v + (1-u)\log(1-v)\right)$ is a binary cross entropy, g_k^+, g_k^- denotes k-th hash code element's estimated likelihood: $g_k^+ = g^+(h_k)$, $g_k^- = g^-(h_k)$, and b_k^+, b_k^- denotes binary likelihood labels which are obtained (refer Figure 2b) as:

$$b_k^+ = \frac{1}{2}\left(\mathrm{sign}(h_k) + 1\right), b_k^- = 1 - b_k^+ \tag{7}$$

As a result, quantization error is reduced by a binary classification loss with the given estimators, allowing to use the merits of cross entropy presented in [2]. Note that, $\mathcal{L}_{bce\text{-}Q}$ is also applied to hash proxies to make them act as continuously relaxed binary codes.

3.3 Training

Total training loss. Suppose we are given a training mini-batch of N_B data points: $\mathcal{X}_B = \{(x_1, \mathrm{y}_1), ..., (x_{N_B}, \mathrm{y}_{N_B})\}$ where each image x_i is assigned a label $\mathrm{y}_i \in \{0,1\}^{N_{cls}}$. Training views are obtained as $\tilde{x}_{Ti} = t_{Ti}(x_i)$ and $\tilde{x}_{Si} = t_{Si}(x_i)$ for all data points, where $t_{Ti} \sim \mathcal{T}_T$ and $t_{Si} \sim \mathcal{T}_S$. Total loss $\mathcal{L}_T$ for DHD is computed with $\mathcal{X}_B$ as:

$$\mathcal{L}_T(\mathcal{X}_B) = \frac{1}{N_B}\sum_{n=1}^{N_B}\left(\mathcal{L}_{HP} + \lambda_1\mathcal{L}_{SdH} + \lambda_2\mathcal{L}_{bce\text{-}Q}\right) \tag{8}$$

where λ_1 and λ_2 are hyper-parameters that balance the influence of the training objectives. The entire DHD framework is trained in an end-to-end fashion.

Multi-label case. In the case of determining semantic similarity between multi-hot labeled images, the previous works [44,51,47] simply checked whether the images share

at least one positive label or not. However, learning with the above similarity has limitations in that the label dependency [9] is ignored. Thus, we aim to capture the intelligence that appears in label dependency by utilizing the Softmax cross entropy with the normalized multi-hot label y. Specifically, y is converted as $\mathrm{y} = \mathrm{y}/\|\mathrm{y}\|_1$ to balance the contribution of each label, and the same $\mathcal{L}_{HP}$ is computed to optimize the deep hashing model for multi-label image retrieval.

4 Experiments

4.1 Setup

Datasets. To evaluate our DHD, we conduct experiments against several conventional and modern methods. Three most popular hashing based retrieval benchmark datasets are explored[6], and we explain the composition of each dataset in Table 1.

Table 1: Description of the image retrieval datasets.

Dataset	# Database	# Train	# Query	N_c
ImageNet [37]	128,503	13,000	5,000	100
NUS-WIDE [10]	149,736	10,500	2,100	21
MS COCO [31]	117,218	10,000	5,000	80

Evaluation metrics. We follow the protocol utilized in deep hashing [4,3,47] to evaluate our approach on both single-labeled and multi-labeled datasets. Specifically, we employ three metrics: 1) mean average precision (**mAP**), 2) precision-recall curves (**PR curves**), and 3) precision with respect to top-M returned image (**P@Top-**M). Regarding mAP score computation, we select the top-M images from the retrieval ranked-list results. The returned images and the query image are considered relevant whether one or more class labels are the same. We set binary code length: hash code dimensionality K as 16, 32, and 64, to examine the performance according to the code size.

4.2 Implementation Details

Data augmentation. Following the works presented in [6], we choose family $\mathcal{T}$ of five image transformations: 1) resized crop, 2) horizontal flip, 3) color jitter, 4) grayscale, and 5) blur, where all of each are sampled uniformly with a given probability and sequentially applied to the inputs. We keep the internal parameters of each transformation equal to [6]. For self-distilled hashing, we configure two groups with $\mathcal{T}$, where the difficult student group is $\mathcal{T}_S = \mathcal{T}$, and the easy teacher group $\mathcal{T}_T$ is configured by scaling all transform occurrence with s_T, which is in the range of $(0, 1]$. We set $\mathcal{T}_T$ as the default for the methods trained without SdH.

[6] The details of each dataset are described in the supplementary material.

Table 2: mean Average Precision (mAP) scores for different bits on three benchmarks.

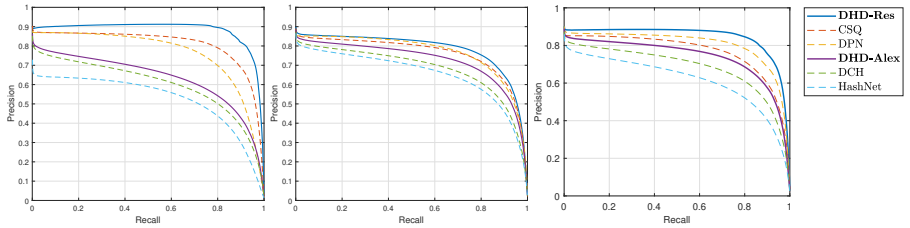
Method	Backbone	ImageNet			NUS-WIDE			MS COCO		
		16-bit	32-bit	64-bit	16-bit	32-bit	64-bit	16-bit	32-bit	64-bit
ITQ [15]	Non-deep	0.266	0.436	0.576	0.435	0.396	0.365	**0.566**	0.562	0.502
SH [43]		0.210	0.329	0.418	0.401	0.421	0.423	0.495	0.507	0.510
KSH [43]		0.160	0.298	0.394	0.394	0.407	0.399	0.521	0.534	0.536
SDH [38]		**0.299**	**0.455**	**0.585**	**0.575**	**0.590**	**0.613**	0.554	**0.564**	**0.580**
CNNH [44]	AlexNet [27]	0.315	0.473	0.596	0.655	0.659	0.647	0.599	0.617	0.620
DNNH [28]		0.353	0.522	0.610	0.703	0.738	0.754	0.644	0.651	0.647
DHN [51]		0.367	0.522	0.627	0.712	0.739	0.751	0.701	0.710	0.735
HashNet [4]		0.425	0.559	0.649	0.720	0.745	0.758	0.685	0.714	0.742
DCH [3]		0.636	0.645	0.656	0.740	0.752	0.763	0.695	0.721	0.748
DHD(Ours)		**0.657**	**0.701**	**0.721**	**0.780**	**0.805**	**0.820**	**0.749**	**0.781**	**0.792**
DPN [14]	ResNet [17]	0.828	0.863	0.872	0.783	0.816	0.838	0.796	0.838	0.861
CSQ [47]		0.851	0.865	0.873	0.810	0.825	0.839	0.750	0.824	0.852
DHD(Ours)		**0.864**	**0.891**	**0.901**	**0.820**	**0.839**	**0.850**	**0.839**	**0.873**	**0.889**
DHD(Ours)	ViT [12]	0.927	0.938	0.944	0.837	0.862	0.870	0.886	0.919	0.939
	DeiT [40]	0.932	0.943	0.948	0.839	0.861	0.867	0.883	0.913	0.925
	SwinT [34]	**0.944**	**0.955**	**0.956**	**0.848**	**0.867**	**0.875**	**0.894**	**0.930**	**0.945**

Experiments. Retrieval experiments are conducted by dividing backbones as: Non-deep, AlexNet [27], ResNet (ResNet50) [17] , and vision Transformers [12,40,34]. For non-deep hashing approaches: ITQ [15], SH [43], KSH [33] and SDH [38], we report the results directly from the latest works [4,3,47] for comparison. We set up the same training environment by leveraging PyTorch framework and the image transformation functions of kornia [13] library for augmentation. We employ Adam optimizer [25] and decay the learning rate with cosine scheduling [35] for training deep hashing methods. Especially for DHD hyper-parameters[7], s_T is set to 0.2 for AlexNet, and 0.5 for other backbones. τ is set by considering N_{cls} as $\{0.2, 0.6, 0.4\}$ for {ImageNet, NUS-WIDE, MS COCO}, respectively. λ_1 and λ_2 are set equal to 0.1 for a balanced contributions each training objective, and σ in $\mathcal{L}_{bce\text{-}Q}$ is set to 0.5 as default.

4.3 Results and Analysis

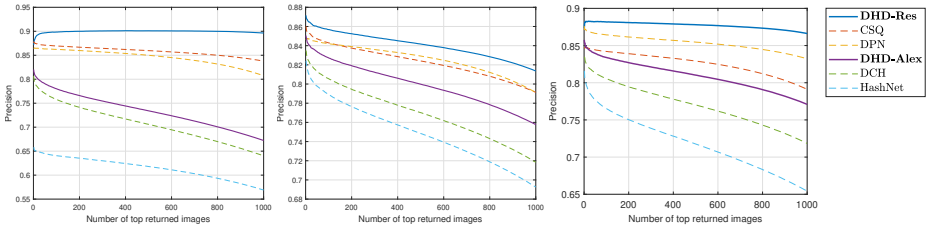
Comparison with others. The mAP scores are calculated by varying the top-M for each dataset as: ImageNet@1000, NUS-WIDE@5000 and MS COCO@5000 to make a fair comparison with previous works [4,3,47]. The results are listed in Table 2, where the highest score for each backbone is shown in bold, and we highlight our DHD method. Among the non-deep hashing methods, SDH shows the best retrieval results by employing supervised label signals in hash function learning. Deep hashing methods generally outperform non-deep hashing ones, since elaborately labeled annotations are fully utilized during training. For ImageNet, NUS-WIDE, and MS COCO, averag-

[7] More details can be found in the supplementary material.



(a) ImageNet (b) NUS-WIDE (c) MS COCO

Fig. 3: Precision-Recall curves on three image datasets with binary codes @ 64-bits.



(a) ImageNet (b) NUS-WIDE (c) MS COCO

Fig. 4: Precision@top-1000 curves on three image datasets with binary codes @ 64-bits.

ing the mAP scores of all bit lengths yields 36.3%p, 33.7%p, and 25.0%p differences between the non-deep and deep methods, respectively.

Notably, our DHD shows the best mAP scores for all datasets in every bit length with every deep backbone architecture. In particular for AlexNet backbone hashing approaches, DHD shows performance improvement of 16.3%p, 7.9%p, and 9.2%p by averaging the mAP scores of all bit lengths in three dataset results orderly, compared to others. To make a comparison with ResNet backbone methods, DHD also achieves 2.7%p, 1.8%p, and 4.7%p higher retrieval scores on average. In line with the recent trend of other computer vision tasks, we *first* introduce Transformer-based image representation learning architectures: ViT [12], DeiT [40], and SwinT [34] to the hashing community and perform retrieval experiments. As reported, when the Transformer is integrated into the DHD framework, it delivers outstanding results for the benchmark image datasets with the increase of 5.8%p, 2.2%p, and 4.8%p, in the same as above, compared to ResNet backbone DHD.

To further investigate the retrieval quality of DHD, we deploy the graph of PR curve and precision for the top 1,000 retrieved images at 64 bits. As shown in Figures 3 and 4, DHD significantly outperforms all the comparison hashing approaches by large margins under these two evaluation metrics. Especially, DHD shows desirable retrieval results in that much higher precision are achieved at lower recall levels, and larger number of top samples are retrieved than all compared methods. These demonstrate the practicality of DHD in real world retrieval cases.

Table 3: mAP scores without (−) or with (+) Self-distilled Hashing (SdH).

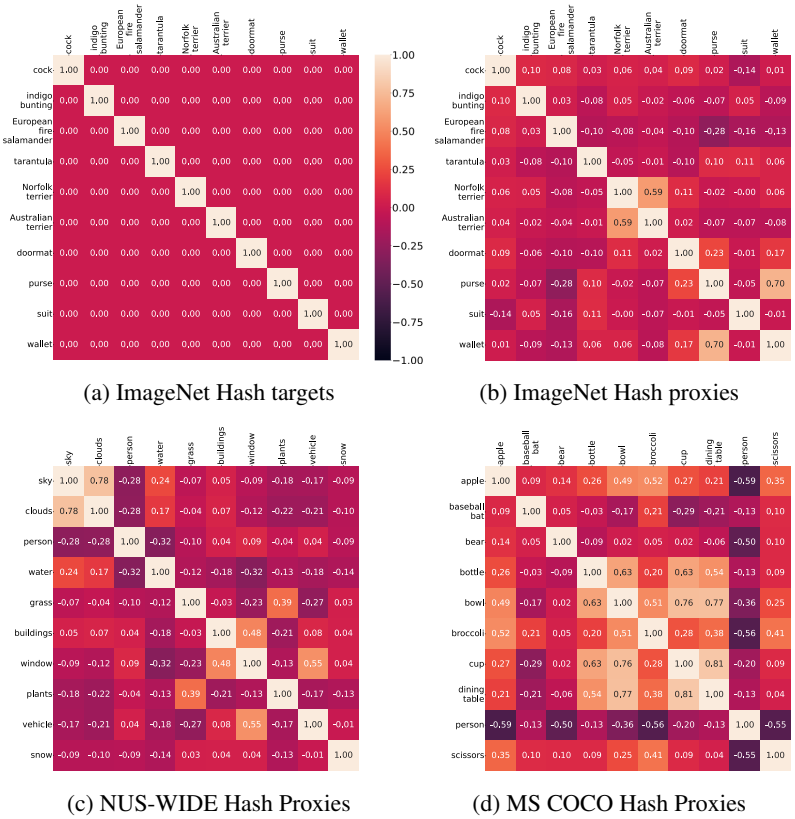
Method	ImageNet				NUS-WIDE				MS COCO			
	− SdH		+ SdH		− SdH		+ SdH		− SdH		+ SdH	
	16-bit	64-bit	16-bit	64-bit	16-bit	64-bit	16-bit	64-bit	16-bit	64-bit	16-bit	64-bit
HashNet [4]	0.337	0.502	0.501	0.661	0.705	0.762	0.745	0.769	0.655	0.727	0.695	0.753
DCH [3]	0.571	0.597	0.640	0.673	0.748	0.767	0.754	0.771	0.669	0.697	0.703	0.746
DPN [14]	0.562	0.656	0.630	0.708	0.753	0.787	0.757	0.801	0.672	0.760	0.710	0.772
CSQ [47]	0.569	0.658	0.634	0.711	0.757	0.793	0.759	0.804	0.670	0.752	0.707	0.765
$\mathcal{L}_{HP}$	0.574	0.660	0.642	0.715	0.759	0.798	0.766	0.812	0.706	0.759	0.725	0.775
$\mathcal{L}_{HP}$ + $\mathcal{L}_{bce\text{-}Q}$	0.583	0.671	**0.657**	**0.721**	0.775	0.806	**0.780**	**0.820**	0.731	0.766	**0.749**	**0.792**

Self-distilled Hashing with other methods and ablations. In order to prove that SdH can be applied to other deep hashing baselines [4,3,14,47], we perform retrieval experiments with AlexNet backbone and show the results in Table 3. With SdH setup, we employ $\mathcal{T}_T$ and $\mathcal{T}_S$ groups to produce input views, and for without SdH setup, we only use $\mathcal{T}_S$ to generate input views. By comparing Table 2 and the results without SdH in Table 3, we can see that the deep hashing model learned with $\mathcal{T}_S$ is inferior to the model learned with $\mathcal{T}_T$. This is because the use of $\mathcal{T}_S$ increases the chances of emerging discrepancy in representation between Hamming and real space. Otherwise, if the model adopts SdH training to utilize both $\mathcal{T}_T$ and $\mathcal{T}_S$, the retrieval performance can be improved since SdH mitigates discrepancy and properly exploits the power of data augmentation. Intending to see the ablation results of our proposals, we compare the retrieval results between $\mathcal{L}_{HP}$ with the others and find that the trainable setting for the hash centers improve the search quality. Moreover, by combining $\mathcal{L}_{HP}$ with $\mathcal{L}_{bce\text{-}Q}$, the performance gain is obtained for all the bit lengths, showing the power of binary cross entropy-based quantization. Finally, the best mAP scores are achieved when both $\mathcal{L}_{HP}$ and $\mathcal{L}_{bce\text{-}Q}$ are integrated with SdH training, confirming the effectiveness of DHD.

Trainable Hash Proxies. In DHD, we employ trainable hash proxies opposed to using predefined orthogonal hash targets [47,14,19], intending to embed detailed class-wise semantic similarity into the hash representation. We visualize[8] the pairwise cosine similarities in Figure 5 using ResNet backbone and 64 bit codes to observe the actual alignment between hash proxies. Since hash targets are generated from a Hadamard matrix, they are orthogonal as shown in Figure 5a. Therefore, the cosine similarity (Hamming distance) between different hash targets are equal, neglecting the semantic relevance between the hash representations of different classes. Moreover for multi-label cases, label dependencies [9] are also ignored whether the contents appear simultaneously in an image or not.

On the other hand, trainable hash proxies are designed to embed semantic similarity by themselves during training. Hence, class-wise relevance can be displayed when we compute pairwise cosine similarities as in Figures 5b to 5d. Specifically for ImageNet, hash proxies of semantically relevant classes have higher similarity, such as *Nor-*

[8] Visualized results with all classes for each dataset are shown in the supplementary material.



(a) ImageNet Hash targets

(b) ImageNet Hash proxies

(c) NUS-WIDE Hash Proxies

(d) MS COCO Hash Proxies

Fig. 5: Pairwise cosine similarities to verify the impact of proxy-based hash representation learning. We utilize (a) pre-defined non-trainable hash targets, and (b-d) proposed trainable hash proxies. For simplicity, we show the results of 10 classes selected.

folk terrier-Australian terrier and *purse-wallet*. Moreover, for multi-label datasets hash proxies of classes that frequently appear together in an image have higher similarity, such as *sky-clouds* and *buildings-window-vehicle* in NUS-WIDE, and *bowl-cup-dining table* in MS COCO. In a nutshell overall, we can confirm that the supervised semantic signals are well guided to represent detailed similarity between the hash proxies, which in turn yields better quality search outcomes.

Insensitivity to transformations. To investigate the sensitivity to transformations, we examine how the binary code shifts when transformed images are fed to ResNet backbone methods, by using ImageNet query set. We measure the average Hamming distance between the untransformed ($s_T = 0$) images' output binary codes and the transformed (s_T in (0, 1]) images' output binary codes, as observed in Figure 6. Here, CSQ [47], DPN [14], and a model learned with $\mathcal{L}_{HP}$ are trained with weakly-transformed $\mathcal{T}_T$, which in result show sensitivity to transformations due to barely used augmentation. When the augmentation is applied ($\mathcal{L}_{HP} + \mathcal{T}_S$), model is improved to be more

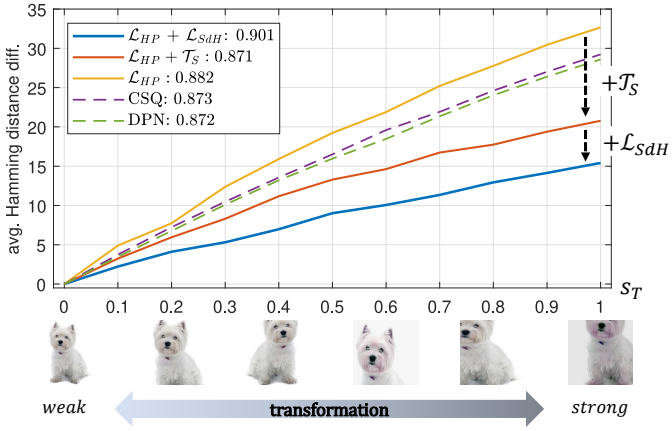


Fig. 6: Average Hamming distance difference between the original and transformed images of ImageNet query set. By varying the s_T, we measure the sensitivity to transformation of ResNet backbone methods, where the numbers in legend indicate mAP. Solid lines present DHD variants, and dotted lines present others. $+\mathcal{T}_S$ denotes strong student augmentation is applied during training. A low slope indicates insensitivity to various transformations, where the blue line (DHD) is the lowest.

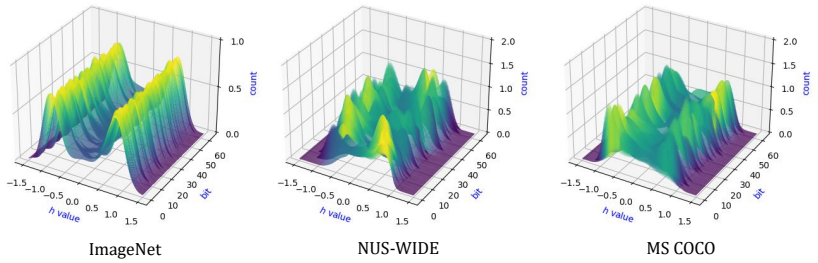
Table 4: mAP scores on unseen deformations.

Deformation	with SdH	without SdH
None	**0.891** (2.3% ↑)	0.871
Cutout	**0.862** (3.7% ↑)	0.827
Dropout	**0.810** (7.9% ↑)	0.765
Zoom in	**0.658** (19.0% ↑)	0.552
Zoom out	**0.816** (1.4% ↑)	0.805
Rotation	**0.856** (2.4% ↑)	0.836
Shearing	**0.842** (2.7% ↑)	0.815
Gaussian noise	**0.768** (10.5% ↑)	0.673

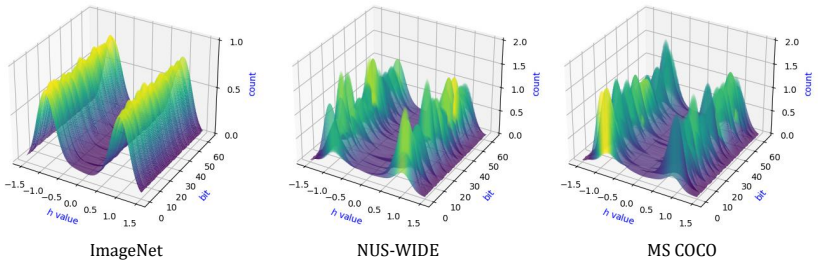
robust to transformations, however, the mAP score decreases due to the discrepancy in representation between Hamming and Real space during training. Otherwise, the combined $\mathcal{L}_{HP} + \mathcal{L}_{SdH}$ (blue line) exhibits the highest robustness while achieving the best mAP score, by minimizing discrepancy in representation during training and successfully exploring the potential of strong augmentation.

Robustness to unseen deformations. To further examine the generalization capacity of DHD, we conduct experiments with unseen (not seen during training) transformations [9] to inputs following the evaluation protocol utilized in [16]. As reported in Table 4, deep hashing model with SdH significantly outperforms the model without SdH at all deformations, showing a performance difference of up to 19% (zoom in). In particular, SdH makes deep hashing model robust to per-pixel deformations such as dropout and Gaussian noise, even though SdH has not included any pixel-level transformations.

[9] Detailed deformation setup is listed in the supplementary material.



(a) without $\mathcal{L}_{bce\text{-}Q}$



(b) with $\mathcal{L}_{bce\text{-}Q}$

Fig. 7: 3D visualized histograms to verify the impact of $\mathcal{L}_{bce\text{-}Q}$. x-axis presents value of hash element h, y-axis presents bit position, and z-axis presents frequency counts.

Quantization. The effect of $\mathcal{L}_{bce\text{-}Q}$ is plotted in Figure 7. We can see that the binary bits are distributed more evenly and binary-like in 7b. This implies that the entropy of bit distribution is much higher when $\mathcal{L}_{bce\text{-}Q}$ is applied, which can show better retrieval accuracy by representing diverse binary codes, as observed and investigated in [30].

5 Conclusion

In this paper, we proposed a novel Self-distilled Hashing (SdH) scheme which is applicable to deep hashing models during training. By maximizing the cosine similarity between hash codes of different views of an image, SdH minimizes the discrepancy in the representation due to augmentation and leads to increase the robustness of retrieval systems. Additionally, we aimed to embed elaborate semantic similarity into the hash codes with a proxy-based learning, and further impose cross entropy-based quantization loss. With all these proposals, we configured Deep Hash Distillation (DHD) framework that yields the state-of-the-art performance on popular deep hashing benchmarks.

Acknowledgement This research was supported in part by NAVER Corporation, the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (2021R1A2C2007220), and the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) [NO.2021-0-01343, Artificial Intelligence Graduate School Program (Seoul National University)].

References

1. Bai, J., Chen, B., Li, Y., Wu, D., Guo, W., Xia, S.t., Yang, E.h.: Targeted attack for deep hashing based retrieval. In: ECCV. pp. 618–634. Springer (2020) 1
2. Boudiaf, M., Rony, J., Ziko, I.M., Granger, E., Pedersoli, M., Piantanida, P., Ayed, I.B.: A unifying mutual information view of metric learning: cross-entropy vs. pairwise losses. In: ECCV. pp. 548–564. Springer (2020) 7
3. Cao, Y., Long, M., Liu, B., Wang, J.: Deep cauchy hashing for hamming space retrieval. In: CVPR. pp. 1229–1237 (2018) 1, 3, 6, 7, 8, 9, 11
4. Cao, Z., Long, M., Wang, J., Yu, P.S.: Hashnet: Deep learning to hash by continuation. In: CVPR. pp. 5608–5617 (2017) 1, 3, 6, 7, 8, 9, 11
5. Caron, M., Touvron, H., Misra, I., Jegou, H., Mairal, J., Bojanowski, P., Joulin, A.: Emerging properties in self-supervised vision transformers. In: ICCV. pp. 9650–9660 (2021) 2, 3, 4, 5
6. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. arXiv preprint arXiv:2002.05709 (2020) 8
7. Chen, W.Y., Liu, Y.C., Kira, Z., Wang, Y.C.F., Huang, J.B.: A closer look at few-shot classification. ICLR (2019) 6
8. Chen, X., He, K.: Exploring simple siamese representation learning. In: CVPR. pp. 15750–15758 (2021) 3, 4, 5
9. Chen, Z.M., Wei, X.S., Wang, P., Guo, Y.: Multi-label image recognition with graph convolutional networks. In: CVPR. pp. 5177–5186 (2019) 8, 11
10. 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: ACM ICMR. pp. 1–9 (2009) 8
11. Cui, Q., Jiang, Q.Y., Wei, X.S., Li, W.J., Yoshie, O.: Exchnet: A unified hashing network for large-scale fine-grained image retrieval. In: ECCV. pp. 189–205. Springer (2020) 1
12. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. In: ICLR (2020) 3, 6, 9, 10
13. E. Riba, e.a.: Kornia: an open source differentiable computer vision library for pytorch. In: WACV (2020), https://arxiv.org/pdf/1910.02190.pdf 9
14. Fan, L., Ng, K., Ju, C., Zhang, T., Chan, C.S.: Deep polarized network for supervised learning of accurate binary hashing codes. In: IJCAI. pp. 825–831 (2020) 3, 4, 6, 9, 11, 12
15. Gong, Y., Lazebnik, S., Gordo, A., Perronnin, F.: Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. PAMI **35**(12), 2916–2929 (2012) 1, 3, 9
16. Gu, G., Ko, B., Kim, H.G.: Proxy synthesis: Learning with synthetic classes for deep metric learning. In: AAAI (2021) 6, 13
17. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778 (2016) 6, 9
18. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015) 4
19. Hoe, J.T., Ng, K.W., Zhang, T., Chan, C.S., Song, Y.Z., Xiang, T.: One loss for all: Deep hashing with a single cosine similarity based learning objective. NeurIPS **34** (2021) 2, 5, 6, 11
20. Jang, Y.K., Cho, N.I.: Deep face image retrieval for cancelable biometric authentication. In: AVSS. pp. 1–8. IEEE (2019) 1
21. Jang, Y.K., Cho, N.I.: Generalized product quantization network for semi-supervised image retrieval. In: CVPR. pp. 3420–3429 (2020) 1, 3
22. Jang, Y.K., Cho, N.I.: Self-supervised product quantization for deep unsupervised image retrieval. In: ICCV. pp. 12085–12094 (2021) 1

23. Jang, Y.K., Jeong, D.j., Lee, S.H., Cho, N.I.: Deep clustering and block hashing network for face image retrieval. In: ACCV. pp. 325–339. Springer (2018) 1, 3, 7
24. Jeong, D.j., Choo, S.K., Seo, W., Cho, N.I.: Classification-based supervised hashing with complementary networks for image search. In: BMVC. p. 74 (2018) 1, 3
25. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR (2015) 9
26. Koch, G., Zemel, R., Salakhutdinov, R., et al.: Siamese neural networks for one-shot image recognition. In: ICML deep learning workshop. vol. 2. Lille (2015) 5
27. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NeurIPS. pp. 1097–1105 (2012) 6, 9
28. Lai, H., Pan, Y., Liu, Y., Yan, S.: Simultaneous feature learning and hash coding with deep neural networks. In: CVPR. pp. 3270–3278 (2015) 9
29. Li, C., Deng, C., Li, N., Liu, W., Gao, X., Tao, D.: Self-supervised adversarial hashing networks for cross-modal retrieval. In: CVPR. pp. 4242–4251 (2018) 1
30. Li, Y., van Gemert, J.: Deep unsupervised image hashing by maximizing bit entropy. In: AAAI (2020) 14
31. 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: European conference on computer vision. pp. 740–755. Springer (2014) 8
32. Liu, S., Wang, Y.: Few-shot learning with online self-distillation. In: ICCV. pp. 1067–1070 (2021) 2
33. Liu, W., Wang, J., Ji, R., Jiang, Y.G., Chang, S.F.: Supervised hashing with kernels. In: CVPR. pp. 2074–2081. IEEE (2012) 1, 2, 3, 5, 9
34. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: ICCV (2021) 3, 6, 9, 10
35. Loshchilov, I., Hutter, F.: Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983 (2016) 9
36. Movshovitz-Attias, Y., Toshev, A., Leung, T.K., Ioffe, S., Singh, S.: No fuss distance metric learning using proxies. In: ICCV. pp. 360–368 (2017) 6
37. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. IJCV **115**(3), 211–252 (2015) 8
38. Shen, F., Shen, C., Liu, W., Tao Shen, H.: Supervised discrete hashing. In: CVPR. pp. 37–45 (2015) 1, 3, 9
39. Shen, Y., Qin, J., Chen, J., Yu, M., Liu, L., Zhu, F., Shen, F., Shao, L.: Auto-encoding twin-bottleneck hashing. In: CVPR. pp. 2818–2827 (2020) 1
40. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: ICML. pp. 10347–10357. PMLR (2021) 2, 3, 4, 6, 9, 10
41. Wang, J., Zhang, T., Sebe, N., Shen, H.T., et al.: A survey on learning to hash. PAMI **40**(4), 769–790 (2017) 1, 3
42. Wang, Z., Zheng, Q., Lu, J., Zhou, J.: Deep hashing with active pairwise supervision. In: ECCV. pp. 522–538. Springer (2020) 1
43. Weiss, Y., Torralba, A., Fergus, R.: Spectral hashing. In: NeurIPS. pp. 1753–1760 (2009) 1, 3, 9
44. Xia, R., Pan, Y., Lai, H., Liu, C., Yan, S.: Supervised hashing for image retrieval via image representation learning. In: AAAI (2014) 1, 3, 7, 9
45. Xu, T.B., Liu, C.L.: Data-distortion guided self-distillation for deep neural networks. In: AAAI. vol. 33, pp. 5565–5572 (2019) 2, 4
46. Yang, E., Liu, T., Deng, C., Liu, W., Tao, D.: Distillhash: Unsupervised deep hashing by distilling data pairs. In: CVPR. pp. 2946–2955 (2019) 1

47. Yuan, L., Wang, T., Zhang, X., Tay, F.E., Jie, Z., Liu, W., Feng, J.: Central similarity quantization for efficient image and video retrieval. In: CVPR. pp. 3083–3092 (2020) 1, 3, 6, 7, 8, 9, 11, 12
48. Yue, Cao, e.a.: Deep quantization network for efficient image retrieval. In: AAAI (2016) 2, 5
49. Yun, S., Park, J., Lee, K., Shin, J.: Regularizing class-wise predictions via self-knowledge distillation. In: CVPR. pp. 13876–13885 (2020) 2, 4
50. Zhou, X., Shen, F., Liu, L., Liu, W., Nie, L., Yang, Y., Shen, H.T.: Graph convolutional network hashing. IEEE Transactions on cybernetics **50**(4), 1460–1472 (2018) 1
51. Zhu, H., Long, M., Wang, J., Cao, Y.: Deep hashing network for efficient similarity retrieval. In: AAAI (2016) 1, 3, 7, 9