Self-supervised Product Quantization for Deep Unsupervised Image Retrieval

Young Kyun Jang   Nam Ik Cho

Department of ECE, INMC, Seoul National University, Seoul Korea

kyun0914@ispl.snu.ac.kr, nicho@snu.ac.kr

Abstract

Supervised deep learning-based hash and vector quantization are enabling fast and large-scale image retrieval systems. By fully exploiting label annotations, they are achieving outstanding retrieval performances compared to the conventional methods. However, it is painstaking to assign labels precisely for a vast amount of training data, and also, the annotation process is error-prone. To tackle these issues, we propose the first deep unsupervised image retrieval method dubbed Self-supervised Product Quantization (SPQ) network, which is label-free and trained in a self-supervised manner. We design a Cross Quantized Contrastive learning strategy that jointly learns codewords and deep visual descriptors by comparing individually transformed images (views). Our method analyzes the image contents to extract descriptive features, allowing us to understand image representations for accurate retrieval. By conducting extensive experiments on benchmarks, we demonstrate that the proposed method yields state-of-the-art results even without supervised pretraining.

1. Introduction

Approximate Nearest Neighbor (ANN) search has received much attention in image retrieval research due to its low storage cost and fast search speed. There are two mainstream approaches in the ANN research, one is *Hashing* [42], and the other is *Vector Quantization* (VQ) [16]. Both methods aim to transform high-dimensional image data into compact binary codes while preserving the semantic similarity, where the difference lies in measuring the distance between the binary codes.

In the case of hashing methods [7, 43, 17, 15, 32], the distance between binary codes is calculated using the Hamming distance, *i.e.,* a simple XOR operation. However, this approach has a limitation that the distance can be represented with only a few distinct values, where the complex distance representation is incapable. To alleviate this issue,
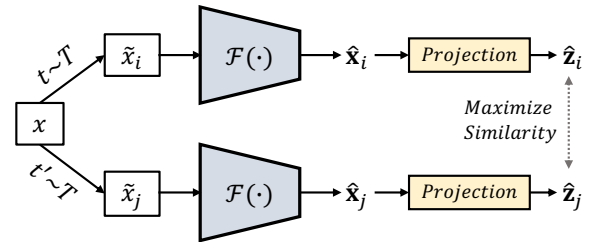
(a) Contrastive Learning
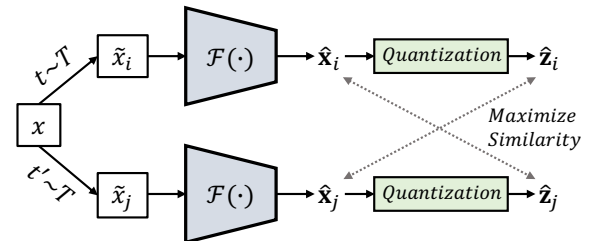
(b) Cross Quantized Contrastive Learning (Ours)

Figure 1. Comparison between (a) contrastive learning and (b) cross quantized contrastive learning. The separately sampled two transformations ($t, t' \sim \mathcal{T}$) are applied on an image x to generate two different views $\tilde{x}_i$ and $\tilde{x}_j$, and corresponding deep descriptor $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{x}}_j$ are obtained from the feature extractor $\mathcal{F}(\cdot)$, respectively. The feature representations in contrastive learning are achieved by comparing the similarity between the projection head outputs $\hat{\mathbf{z}}_i$ and $\hat{\mathbf{z}}_j$. Instead of projection, we introduce the quantization head, which collects codebooks of product quantization. By maximizing cross-similarity between the deep descriptor of one view and the product quantized descriptor of the other, both codewords and deep descriptors are jointly trained to contain discriminative image content representations.

VQ-based methods [23, 13, 24, 2, 48, 3, 49] have been proposed, exploiting quantized real-valued vectors in distance measurement instead. Among these, *Product Quantization* (PQ) [23] is one of the best methods, delivering the retrieval results very fast and accurately.

The essence of PQ is to decompose a high-dimensional space of feature vectors (image descriptors) into a Cartesian product of several subspaces. Then, each of the im-

age descriptors is divided into several subvectors according to the subspaces, and the subvectors are clustered to form centroids. As a result, *Codebook* of each subspace is configured with corresponding centroids (*codewords*), which are regarded as quantized representations of the images. The distance between two different binary codes in the PQ scheme is asymmetrically approximated by utilizing real-valued codewords with look-up table, resulting in richer distance representations than the hashing.

Recently, supervised deep hashing methods [44, 5, 22, 29, 47] show promising results for large-scale image retrieval systems. However, since binary hash codes cannot be directly applied to learn deep continuous representations, performance degradation is inevitable compared to the retrieval using real vectors. To address this problem, quantization-based deep image retrieval approaches have been proposed in [20, 4, 46, 31, 26, 21]. By introducing differentiable quantization methods on continuous deep image feature vectors (deep descriptors), direct learning of deep representations is allowed in the real-valued space.

Although deep supervised image retrieval systems provide outstanding performances, they need expensive training data annotations. Hence, deep unsupervised hashing methods have also been proposed [30, 11, 39, 50, 41, 14, 36, 45, 37], which investigate the image similarity to discover semantically distinguishable binary codes without annotations. However, while quantization-based methods have advantages over hashing-based ones, only limited studies exist that adopt quantization for deep unsupervised retrieval. For example, [35] employed pre-extracted visual descriptors instead of images for the unsupervised quantization.

In this paper, we propose the first *unsupervised* end-to-end deep quantization-based image retrieval method; *Self-supervised Product Quantization* (SPQ) network, which jointly learns the feature extractor and the codewords. As shown in Figure 1, the main idea of SPQ is based on self-supervised contrastive learning [8, 40, 6]. We regard that two different "views" (individually transformed outputs) of a single image are correlated, and conversely, the views generated from other images are uncorrelated. To train PQ codewords, we introduce *Cross Quantized Contrastive learning*, which maximizes the cross-similarity between the correlated deep descriptor and the product quantized descriptor. This strategy leads both deep descriptors and PQ codewords to become discriminative, allowing the SPQ framework to achieve high retrieval accuracy.

To demonstrate the efficiency of our proposal, we conduct experiments under various training conditions. Specifically, unlike previous methods that utilize pretrained model weights learned from a large labeled dataset, we conduct experiments with "truly" unsupervised settings where human supervision is excluded. Despite the absence of label information, SPQ achieves state-of-the-art performance.

The contributions of our work are summarized as:

- To the best of our knowledge, SPQ is the first deep unsupervised quantization-based image retrieval scheme, where both feature extraction and quantization are included in a single framework and trained in a self-supervised fashion.

- By introducing cross quantized contrastive learning strategy, the deep descriptors and the PQ codewords are jointly learned from two different views, delivering discriminative representations to obtain high retrieval scores.

- Extensive experiments on fast image retrieval protocol datasets verify that our SPQ shows state-of-the-art retrieval performance even for the truly unsupervised settings.

2. Related Works

This section categorizes image retrieval algorithms regarding whether or not deep learning is utilized (conventional versus deep methods) and briefly explains the approaches. For a more comprehensive understanding, refer to a survey paper [42].

Conventional methods. One of the most common strategies for fast image retrieval is hashing. For some examples, Locality Sensitivity Hashing (LSH) [7] employed random linear projections to hash. Spectral Hashing (SH) [43] and Discrete Graph Hashing (DGH) [32] exploited graph-based approaches to preserve data similarity of the original feature space. K-means Hashing (KMH) [17] and Iterative Quantization (ITQ) [15] focused on minimizing quantization errors that occur when mapping the original feature to discrete binary codes.

Another fast image retrieval strategy is vector quantization. There are Product Quantization (PQ) [23] and its improved variants; Optimized PQ (OPQ) [13], Locally Optimized PQ (LOPQ) [24], and methods with different quantizers, such as Additive [2], Composite [48], Tree [3], and Sparse Composite Quantizers [49]. Our SPQ belongs to the PQ family, where the deep feature data space is divided into several disjoint subspaces. The divided deep subvectors are then trained with our proposed loss function to find the optimal codewords.

Deep methods. Supervised deep convolutional neural network (CNN)-based hashing approaches [44, 5, 22, 29, 47] have shown superior performance in many image retrieval tasks. There are also quantization-based deep image retrieval methods [4, 26], which use pretrained CNNs and fine-tune the network to train robust codewords together. For improvement, the metric learning schemes are applied in [46, 31, 21] to learn codewords and deep representations together with the pairwise semantic similarity. Note that we
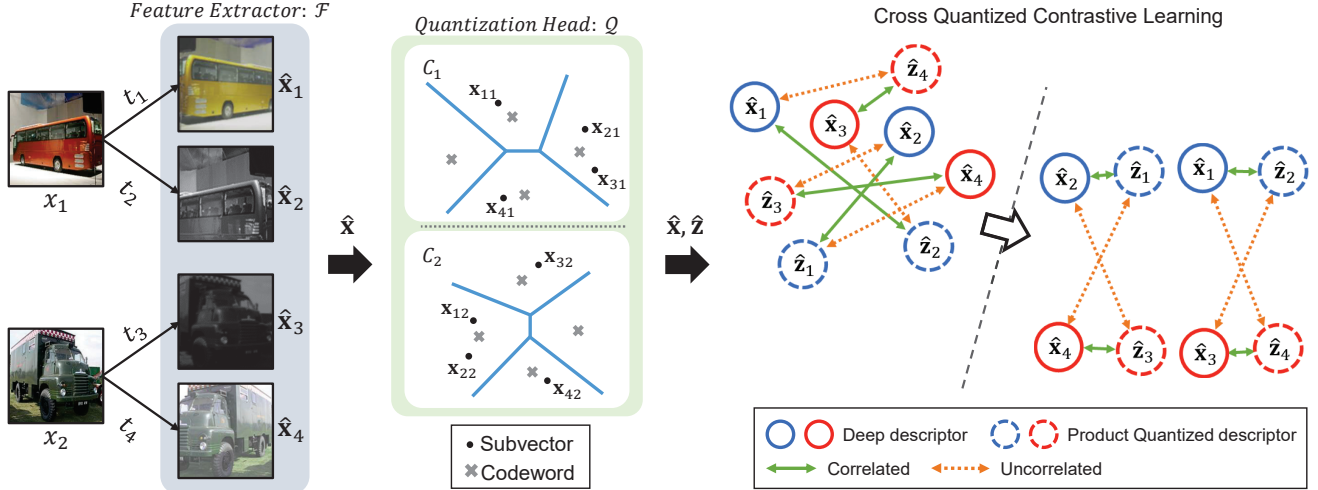
Figure 2. An illustration of feature extraction, quantization, and training procedure in SPQ. Randomly sampled data augmentation techniques ($t_n \sim T$) are applied on x_1 and x_2 to produce transformed images (different views). There are two trainable components; (1) CNN-based feature extractor $\mathcal{F}$, and (2) quantization head $\mathcal{Q}$, which collects multiple codebooks to conduct product quantization. For example, we set up two codebooks C_1 and C_2, and illustrate 2D conceptual Voronoi diagram in $\mathcal{Q}$. The original feature space of deep descriptor (feature vector $\hat{\mathbf{x}}_n \in R^D$) is divided into two subspaces and generates subvectors; $\mathbf{x}_{nm}$ where $m = \{1, 2\}$ and $\mathbf{x}_{nm} \in R^{D/2}$. By employing soft quantizer $q_m(\cdot)$ on each $\mathbf{x}_{nm}$, the sub-quantized descriptor $\mathbf{z}_{nm} = q_m(\mathbf{x}_{nm})$ is approximated with the combination of the codewords. Notably, subvectors representing similar features are allocated to the same codeword. The output product quantized descriptor $\hat{\mathbf{z}}_n \in R^D$ is obtained by concatenating the sub-quantized descriptors along the D-dimension. For better understanding, we paint the feature representations related to x_1 in blue, and x_2 in red. Taking into account the cross-similarity between $\hat{\mathbf{x}}$ and $\hat{\mathbf{z}}$ as: $\hat{\mathbf{x}}_1 \leftrightarrow \{\hat{\mathbf{z}}_2, \hat{\mathbf{z}}_4\}$, $\hat{\mathbf{x}}_2 \leftrightarrow \{\hat{\mathbf{z}}_1, \hat{\mathbf{z}}_3\}$, $\hat{\mathbf{x}}_3 \leftrightarrow \{\hat{\mathbf{z}}_2, \hat{\mathbf{z}}_4\}$, and $\hat{\mathbf{x}}_4 \leftrightarrow \{\hat{\mathbf{z}}_1, \hat{\mathbf{z}}_3\}$, the network is trained to understand the discriminative image contents, while simultaneously collecting frequently occurring local patterns into the codewords.

also utilize a type of metric learning, *i.e.,* contrastive learning; however, our method requires no label information in learning the codewords.

Regarding unsupervised deep image retrieval, most works are based on hashing. To be specific, generative mechanisms are utilized in [11, 39, 50, 14], and graph-based techniques are employed in [36, 37]. Notably, DeepBit [30] has a similar concept to SPQ in that the distance between the transformed image and the original one is minimized. However, the hash code representation has a limitation in that only a simple rotational transformation is exploited. In terms of deep quantization, there only exists a study dubbed Unsupervised Neural Quantization (UNQ) [35], which uses pre-extracted visual descriptors instead of employing the image itself to find the codewords.

To improve the quality of image descriptors and codewords for unsupervised deep PQ-based retrieval, we configure SPQ with a feature extractor to explore the entire image information. Then, we jointly learn every component of SPQ in a self-supervised fashion. Similar to [8, 40, 6], the full knowledge of the dataset is augmented with several transformations such as crop and resize, flip, color distortion, and Gaussian blurring. By cross-contrasting differently augmented images, both image descriptors and codewords become discriminative to achieve a high retrieval score.

3. Self-supervised Product Quantization

3.1. Overall Framework

The goal of an image retrieval model is to learn a mapping $\mathcal{R} : x \mapsto \mathbf{b}$ where $\mathcal{R}$ denotes the overall system, x is an image included in a dataset $\mathcal{X} = \{x_n\}_{n=1}^N$ of N training samples, and $\hat{\mathbf{b}}$ is a B-bits binary code $\hat{\mathbf{b}} \in \{0, 1\}^B$. As shown in Figure 2, $\mathcal{R}$ of the SPQ contains a deep CNN-based feature extractor $\mathcal{F}(x; \theta_{\mathcal{F}})$ which outputs a compact deep descriptor (feature vector) $\mathbf{x} \in R^D$. Any CNN architecture can be exploited as a feature extractor, as long as it can handle fully connected layer, e.g. AlexNet [28], VGG [38], or ResNet [18]. We configure the baseline network architecture with ResNet50 that generally shows outstanding performance in image representation learning, and details are reported in section 4.2.

Regarding the quantization for fast image retrieval, $\mathcal{R}$ employs M codebooks in the quantization head $\mathcal{Q}(\hat{\mathbf{x}}; \theta_{\mathcal{Q}})$ of $\{C_1, ..., C_M\} \subset \mathcal{Q}$, where C_i consists of K codewords $\mathbf{c} \in R^{D/M}$ as $C_m = \{\mathbf{c}_{m1}, ..., \mathbf{c}_{mK}\}$. PQ is conducted in $\mathcal{Q}$ by dividing the deep feature space into the Cartesian product of multiple subspaces. Every codebook of corresponding subspace exhibits several distinctive characteristics representing the image dataset $\mathcal{X}$. Each codeword belonging to the codebook infers a clustered centroid of a divided deep descriptor, which aims to hold a local pattern

that frequently occurs. During quantization, similar properties between images are shared by being assigned to the same codeword, whereas distinguishable features have different codewords. As a result, various distance representations for efficient image retrieval are achieved.

3.2. Self-supervised Training

For better understanding, we briefly describe the training scheme of SPQ in Algorithm 1, where $\theta_{\mathcal{F}}$ and $\theta_{\mathcal{Q}}$ denote trainable parameters of feature extractor and quantization head, respectively, and γ is a learning rate. In this case, $\theta_{\mathcal{Q}}$ represents a collection of codebooks. The training scheme and quantization process of SPQ are detailed below.

First of all, to conduct deep learning with $\mathcal{F}$ and $\mathcal{Q}$ in an end-to-end manner, and make the whole codewords contribute to training, we need to solve the infeasible derivative calculation of hard assignment quantization. For this, following the method in [46], we introduce soft quantization on the quantization head with the soft quantizer $q_m(\cdot)$ as:

$$\mathbf{z}_{nm} = \sum_k^K \frac{\exp(-\|\mathbf{x}_{nm} - \mathbf{c}_{mk}\|_2^2/\tau_q)}{\sum_{k'}^K \exp(-\|\mathbf{x}_{nm} - \mathbf{c}_{mk'}\|_2^2/\tau_q)} \mathbf{c}_{mk} \quad (1)$$

where τ_q is a non-negative temperature parameter that scales the input of the softmax, and $\|\cdot\|_2^2$ denotes the squared Euclidean distance to measure the similarity between inputs. In this fashion, the sub-quantized descriptor $\mathbf{z}_{nm} = q_m(\mathbf{x}_{nm}; \tau_q, C_m)$ can be regarded as an exponential weighted sum of the codewords that belong to C_m. Note that the entire codewords in the codebook are utilized to approximate the quantized output, where the closest codeword contributes the most.

Besides, unlike previous deep PQ approaches [46, 21], we exclude intra-normalization [1] which is known to minimize the impact of burst visual features when concatenating sub-quantized descriptors to obtain the whole product quantized descriptor $\hat{\mathbf{z}}$. Since our SPQ is trained without any human supervision, which assists in finding distinct features, we focus on catching dominant visual features rather than balancing the influence of every codebook.

To learn the deep descriptors and the codewords together, we propose *cross quantized contrastive learning* scheme. Inspired by contrastive learning [8, 40, 6], we attempt to compare the deep descriptors and the product quantized descriptors of various views (transformed images). As observed in Figure 2, the deep descriptor and the product quantized descriptor are treated as correlated if the views are originated from the same image, whereas uncorrelated if the views are originated from the different images. Note that, to increase the generalization capacity of the codewords, the correlation between the deep descriptor and the quantized descriptor of itself ($\hat{\mathbf{x}}_n$ and $\hat{\mathbf{z}}_n$) is ignored. This

is because the contribution of other codewords decreases when the agreement between the subvector and the nearest codeword is maximized.

For a given mini-batch of size N_B, we randomly sample N_B examples from the database $\mathcal{X}$ and apply a random combination of augmentation techniques to each image twice to generate $2N_B$ data points (views). Inspired from [9, 8, 6], we take into account that two separate views of the same image $(\tilde{x}_i, \tilde{x}_j)$ are correlated, and the other $2(N_B - 1)$ views originating from different images within a mini-batch are uncorrelated. On this assumption, we design a cross quantized contrastive loss function to learn the correlated pair of examples (i, j) as:

$$\ell_{(i,j)} = -\log \frac{\exp\left(\mathcal{S}(i,j)/\tau_{cqc}\right)}{\sum_{n=1}^{N_B} \mathbb{1}_{[n' \neq j]} \exp\left(\mathcal{S}(i,n')/\tau_{cqc}\right)} \quad (2)$$

where $n' = \begin{cases} 2n-1 & \text{if } j \text{ is odd} \\ 2n & \text{else} \end{cases}$, $\mathcal{S}(i,j)$ denotes a cosine similarity between $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{z}}_j$, τ_{cqc} is a non-negative temperature parameter, and $\mathbb{1}_{[n' \neq j]} \in \{0, 1\}$ is an indicator that evaluates to 1 iff $n' \neq j$. Notably, to reduce redundancy between $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{z}}_i$ which are similar to each other, the loss is computed for the half of the uncorrelated samples in the batch. The cosine similarity is used as a distance measure to avoid the norm deviations between $\hat{\mathbf{x}}$ and $\hat{\mathbf{z}}$.

Algorithm 1 SPQ's main learning algorithm.

Input: Trainable parameters : $\theta_{\mathcal{F}}, \theta_{\mathcal{Q}}$, batch size N_B
1: **for** sampled mini-batch $\{x_n\}_{n=1}^{N_B}$ **do**
2:     **for** n in $\{1,\dots,N_B\}$ **do**
3:         draw two transformations $t_{2n-1} \sim \mathcal{T}, t_{2n} \sim \mathcal{T}$
4:         $\tilde{x}_{2n-1} \leftarrow t_{2n-1}(x_n)$
5:         $\tilde{x}_{2n} \leftarrow t_{2n}(x_n)$
6:         $\hat{\mathbf{x}}_{2n-1}, \hat{\mathbf{x}}_{2n} = \mathcal{F}(\tilde{x}_{2n-1}), \mathcal{F}(\tilde{x}_{2n})$
7:         $\hat{\mathbf{z}}_{2n-1}, \hat{\mathbf{z}}_{2n} = \mathcal{Q}(\hat{\mathbf{x}}_{2n-1}), \mathcal{Q}(\hat{\mathbf{x}}_{2n})$
8:     **end for**
9:     **for** i in $\{1,\dots,N_B\}$ and j in $\{1,\dots,N_B\}$ **do**
10:       $\mathcal{S}(2i-1, 2j) = \hat{\mathbf{x}}_{2i-1}^T \hat{\mathbf{z}}_{2j}/(\|\hat{\mathbf{x}}_{2i-1}\|\|\hat{\mathbf{z}}_{2j}\|)$
11:       $\mathcal{S}(2i, 2j-1) = \hat{\mathbf{x}}_{2i}^T \hat{\mathbf{z}}_{2j-1}/(\|\hat{\mathbf{x}}_{2i}\|\|\hat{\mathbf{z}}_{2j-1}\|)$
12:     **end for**
13:     $\mathcal{L}_{cqc} = \frac{1}{2N_B} \sum_{n=1}^{N_B} \left(\ell_{(2n-1,2n)} + \ell_{(2n,2n-1)}\right)$
14:     $\theta_{\mathcal{F}} \leftarrow \theta_{\mathcal{F}} - \gamma \frac{\partial \mathcal{L}_{cqc}}{\partial \theta_{\mathcal{F}}}$
15:     $\theta_{\mathcal{Q}} \leftarrow \theta_{\mathcal{Q}} - \gamma \frac{\partial \mathcal{L}_{cqc}}{\partial \theta_{\mathcal{Q}}}$
16: **end for**
Output: Updated $\theta_{\mathcal{F}}, \theta_{\mathcal{Q}}$

Concerning the data augmentation for generating various views, we employ five popular techniques: (1) resized crop to treat local, global, and adjacent views, (2) horizontal flip to handle mirrored inputs, (3) color jitter to deal with color

distortions, (4) grayscale to focus more on intensity, and (5) Gaussian blur to cope with noise in the image. The default setup is directly taken from [8], where all transformations are randomly applied in a sequential manner (1-5). Exceptionally, we modify color jitter strength as 0.5 to fit in SPQ, following the empirical observation. In the end, SPQ is capable of interpreting contents in the image by contrasting different views of images in a self-supervised way.

3.3. Retrieval

Image retrieval is performed in two steps, similar to PQ [23]. First, the retrieval database consisting of binary codes is configured with a dataset $\mathcal{X}_g = \{x_n\}_{n=1}^{N_g}$ of N_g gallery images. By employing $\mathcal{F}$, the deep descriptor $\hat{\mathbf{x}}_n$ is obtained from x_n and divided into M equal-length subvectors as $\hat{\mathbf{x}}_n = [\mathbf{x}_{n1}, ..., \mathbf{x}_{nM}]$. Then, the nearest codeword of each subvector $\mathbf{x}_{nm}$ is searched by computing the squared Euclidean distance ($\|\cdot\|_2^2$) between the subvector and every codeword in the codebook C_m. Then, the index of the nearest codeword k^* is formatted as a binary code to generate a sub-binary code $\mathbf{b}_{nm}$. Finally, all the sub-binary codes are concatenated to generate the $M \cdot \log_2(K)$-bits binary code $\hat{\mathbf{b}}_n$, where $\hat{\mathbf{b}}_n = [\mathbf{b}_{n1}, ..., \mathbf{b}_{nM}]$. We repeat this process for all gallery images to build a binary encoded retrieval database.

Moving on to the retrieval stage, we apply the same $\mathcal{F}$ and dividing process on a query image x_q to extract $\hat{\mathbf{x}}_q$ and a set of its subvectors as $\hat{\mathbf{x}}_q = [\mathbf{x}_{q1}, ..., \mathbf{x}_{qM}]$. The Euclidean distance is utilized to measure the similarity between the subvectors and every codeword of all codebooks to construct a pre-computed look-up table. The distance calculation between the query and the gallery is asymmetrically approximated and accelerated by summing up the look-up results.

4. Experiments

4.1. Datasets

To evaluate the performance of SPQ, we conduct comprehensive experiments on three public benchmark datasets, following experimental protocols in recent unsupervised deep image retrieval methods [50, 45, 37].
CIFAR-10 [27] contains 60,000 images with the size of 32×32 in 10 class labels, and each class has 6,000 images. We select 5,000 images per class as a training set, 100 images per class as a query set. The entire training set of 50,000 images are utilized to build a retrieval database.
FLICKR25K [19] consists 25,000 images with various resolutions collected from the Flickr website. Every image is manually annotated with at least one of the 24 semantic labels. We randomly take 2,000 images as a query set and employ the remaining 23,000 images to build a retrieval database, of which 5,000 images are utilized for training.

NUS-WIDE [10] has nearly 270,000 images with various resolutions in 81 unique labels, where each image belongs to one or more labels. We pick out images containing the 21 most frequent categories to perform experiments with a total of 169,643. We randomly choose a total of 10,500 images as a training set with each category being at least 500, a total of 2,100 images as a query set with each category being at least 100, and the rest images as a retrieval database.

Table 1. Composition of three benchmark datasets.

Dataset	# Train	# Query	# Retrieval	# Class
CIFAR-10	50,000	10,000	50,000	10
FLICKR25K	5,000	2,000	23,000	24
NUS-WIDE	10,500	2,100	157,043	21

4.2. Experimental Settings

Evaluation Metrics. We employ mean Average Precision (mAP) to evaluate the retrieval performance. Specifically, in the case of multi-label image retrieval on FLICKR25K and NUS-WIDE dataset, it is considered relevant even if only one of the labels matches. We vary the number of bits allocated to the binary code as $\{16, 32, 64\}$ to measure the mAP scores of the retrieval approaches, mAP@1,000 for CIFAR-10 dataset and mAP@5,000 for FLICKR25K and NUS-WIDE datasets, following the evaluation method in [37, 45]. In addition, by employing 64-bits hash codes of different algorithms, we draw Precision-Recall curves (PR) to compare the precisions at different recall levels and report Precision curves with respect to 1,000 top returned samples (P@1,000) to contrast the ratio of results retrieved correctly.
Implementation details. There are three baseline approaches that we categorize to make a comparison. Specifically, (1) shallow methods without deep learning, based on hashing: LSH [7], SH [43], ITQ [15], and based on product quantization: PQ [23], OPQ [13] LOPQ [24], (2) deep semi-unsupervised methods: DeepBit [30], Greedy-Hash [41], DVB [36], DistillHash [45], TBH [37], and (3) deep truly unsupervised methods: SGH [11], HashGAN [14], BinGAN [50], BGAN [39]. The terms "semi" and "truly" indicate whether the pretrained model weights are utilized or not. Both semi and truly training conditions can be applied to SPQ; however, we take the truly unsupervised model that has the advantage of not requiring human supervision as the baseline.

To evaluate the shallow and deep semi-unsupervised methods, we employ ImageNet pretrained model weights of AlexNet [28] or VGG16 [38] to utilize fc_7 features, following the experimental settings of [45, 36, 37]. Since those models take only fixed-size inputs, we need to resize all images to 224×224, by upscaling the small images and downsampling the large ones. In the case of evaluating deep truly unsupervised methods, including SPQ, the same re-

Table 2. mAP scores of different retrieval methods on three benchmark datasets.

Method	CIFAR-10			FLICKR25K			NUS-WIDE		
	16-bits	32-bits	64-bits	16-bits	32-bits	64-bits	16-bits	32-bits	64-bits
Shallow Methods without Deep Learning									
LSH [7]	0.132	0.158	0.167	0.583	0.589	0.593	0.432	0.441	0.443
SH [43]	0.272	0.285	0.300	0.591	0.592	0.602	0.510	0.512	0.518
ITQ [15]	0.305	0.325	0.349	0.610	0.622	0.624	0.627	0.645	0.664
PQ [23]	0.237	0.259	0.272	0.601	0.612	0.626	0.452	0.464	0.479
OPQ [13]	0.297	0.314	0.323	0.620	0.626	0.629	0.565	0.579	0.598
LOPQ [24]	0.314	0.320	0.355	0.614	0.634	0.635	0.620	0.655	0.670
Deep Semi-unsupervised Methods									
DeepBit [30]	0.220	0.249	0.277	0.593	0.593	0.620	0.454	0.463	0.477
GreedyHash [41]	0.448	0.473	0.501	0.689	0.699	0.701	0.633	0.691	0.731
DVB [36]	0.403	0.422	0.446	0.614	0.655	0.658	0.677	0.632	0.665
DistillHash [45]	0.454	0.469	0.489	0.696	0.706	0.708	0.667	0.675	0.677
TBH [37]	0.532	0.573	0.578	0.702	0.714	0.720	0.717	0.725	0.735
Deep truly unsupervised Methods									
SGH [11]	0.435	0.437	0.433	0.616	0.628	0.625	0.593	0.590	0.607
HashGAN [14]	0.447	0.463	0.481	-	-	-	-	-	-
BinGAN [50]	0.476	0.512	0.520	0.663	0.679	0.688	0.654	0.709	0.713
BGAN [39]	0.525	0.531	0.562	0.671	0.686	0.695	0.684	0.714	0.730
SPQ (Ours)	**0.768**	**0.793**	**0.812**	**0.757**	**0.769**	**0.778**	**0.766**	**0.774**	**0.785**

sized images of FLICKR25K and NUS-WIDE datasets are used for simplicity, and the original resolution images of CIFAR-10 are used to reduce computational load.

Our implementation of SPQ is based on PyTorch with NVIDIA Tesla V100 32GB Tensor Core GPU. Following the observations in recent self-supervised learning studies [8, 6], we set the baseline network architecture as a standard ResNet50 [18] for FLICKR25K and NUS-WIDE datasets. In the case of the CIFAR-10 dataset with much smaller images, we set the baseline as a standard ResNet18 [18], and modify the number of filters as same as ResNet50.

For network training, we adopt Adam [25] and decay the learning rate with the cosine scheduling without restarts [33] and set the batch size N_B as 256. We fix the dimension of the subvector x and the codeword c to $D/M = 16$, and also the number of codewords to $K = 2^4$. Consequently, the number codebooks M is changed to $\{4, 8, 16\}$ because $M \cdot \log_2(K)$-bits are needed to obtain $\{16, 32, 64\}$-bits binary code. The temperature parameter τ_q and τ_{cqc} are set as 5 and 0.5, respectively. Data augmentation is operated with Kornia [12] library, and each transformation is applied with the same probability as the settings in [8].

4.3. Results

The mAP results on three different image retrieval datasets are listed in Table 2, showing that SPQ substantially outperforms all the compared methods in every bit-length. Additionally, referring to Figures 3 and 4, SPQ is demonstrated to be the most desirable retrieval system.

First of all, compared to the best shallow method LOPQ [24], SPQ reveals a performance improvement of more than 46%p, 13%p, and 11.6%p in the average mAP on CIFAR-10, FLICKR25K, and NUS-WIDE, respectively. The reason for the more pronounced difference for CIFAR-10 is because the shallow methods involve an unnecessary upscaling process to utilize the ImageNet pretrained deep features. SPQ has an advantage over shallow methods in that various and suitable neural architectures can be accommodated for feature extraction and end-to-end learning.

Second, in contrast to the best deep semi-unsupervised method TBH [37], SPQ yields 23%p, 4.6%p, and 3.9%p higher average mAP scores on CIFAR-10, FLICKR25K, and NUS-WIDE, respectively. Even in the absence of prior information such as pretrained model weights, SPQ well distinguishes the contents within the image by comparing multiple views of training samples.
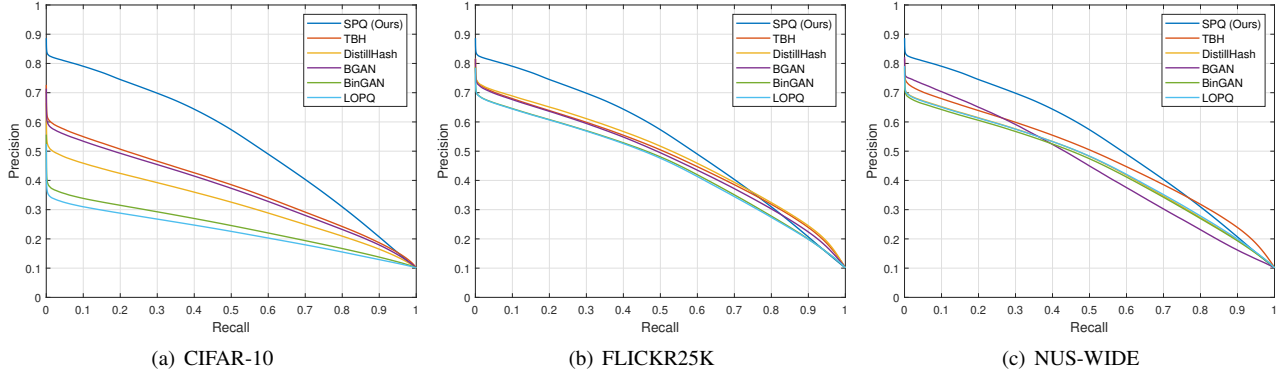
(a) CIFAR-10   (b) FLICKR25K   (c) NUS-WIDE

Figure 3. Precision-Recall curves on three benchmark datasets with binary codes @ 64-bits.

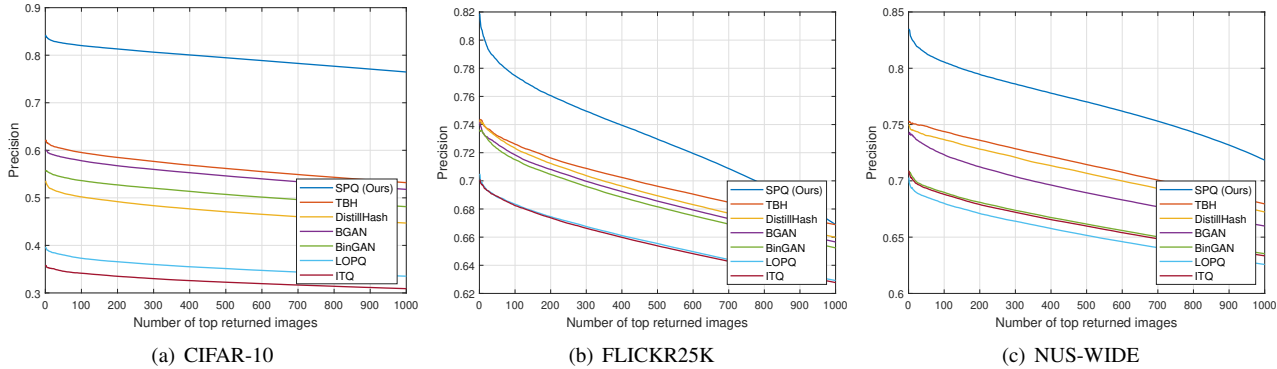(a) CIFAR-10   (b) FLICKR25K   (c) NUS-WIDE

Figure 4. Precision@top-1000 curves on three benchmark datasets with binary codes @ 64-bits.

Lastly, even with the truly unsupervised setup, SPQ achieves state-of-the-art retrieval accuracy. Specifically, unlike previous hashing-based truly unsupervised methods, SPQ introduces differentiable product quantization to the unsupervised image retrieval system for the first time. By considering cross-similarity between different views in a self-supervised way, deep descriptors and codewords are allowed to be discriminative.

4.4. Empirical Analysis

4.4.1 Ablation Study

We configure five variants of SPQ to investigate: (1) *SPQ-C* that replaces cross-quantized contrastive learning with contrastive learning by comparing $\hat{z}_i$ and $\hat{z}_j$, (2) *SPQ-H*, employing hard quantization instead of soft quantization, (3) *SPQ-Q* that employs standard vector quantization, which does not divide the feature space and directly utilize entire feature vector to build the codebook, (4) *SPQ-S* that exploits pretrained model weights to conduct deep semi-unsupervised image retrieval, and (5)*SPQ-V*, utilizing VGG16 network architecture as the baseline.

As reported in Table 3, we can observe that each component of SPQ contributes sufficiently to performance improvement. Comparison with SPQ-C confirms that consid-

Table 3. mAP scores of the previous best method, SPQ and its variants on three benchmark datasets @ 32-bits.

Method	CIFAR-10	FLICKR25K	NUS-WIDE
TBH [37]	0.573	0.714	0.725
SPQ-C	0.763	0.751	0.756
SPQ-H	0.745	0.736	0.742
SPQ-Q	0.734	0.733	0.738
SPQ-S	**0.814**	**0.781**	**0.788**
SPQ-V	0.761	0.749	0.753
SPQ	0.793	0.769	0.774

ering cross-similarity rather than comparing quantized outputs delivers more efficient image retrieval results. From the results of SPQ-H, we find that soft quantization is more suitable for learning codewords. The retrieval outcomes with SPQ-Q, which shows the biggest performance gap with SPQ, explain that product quantization leads to accomplishing precise search results by increasing the amount of distance representation. Notably, SPQ-S, which utilizes ImageNet pretrained model weights for network initialization, outperforms truly unsupervised SPQ. In this observation, we can see that although SPQ demonstrates the best
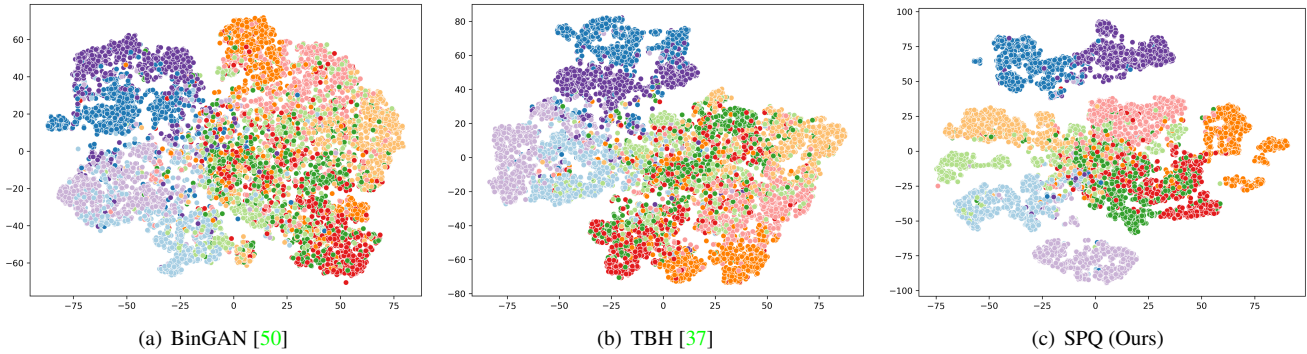
| (a) BinGAN [50] | (b) TBH [37] | (c) SPQ (Ours) |

Figure 5. t-SNE visualization of deep representations learned by BinGAN, TBH, and SPQ on CIFAR-10 query set respectively.

(a) τ_q vs. color jitter strength.　(b) τ_{cqc} vs. color jitter strength.
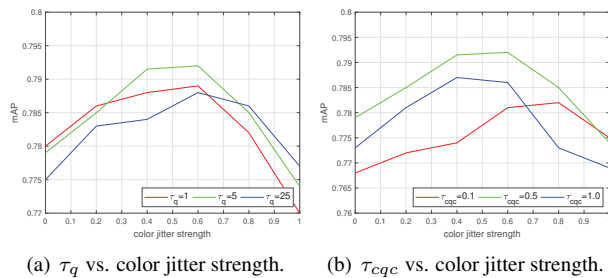
Figure 6. Sensitivity investigation of hyper-parameters according to the color jitter strengths on CIFAR-10 @ 32-bits.

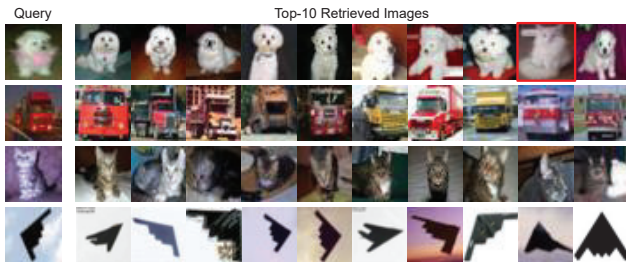Figure 7. SPQ retrieval results on CIFAR-10 @ 32-bits.

retrieval accuracy without any human guidance, better results can be obtained with some label information. Although SPQ-V is inferior to ResNet-based SPQ, its performance still surpasses existing state-of-the-art retrieval algorithms, which proves the excellence of the PQ-based self-supervised learning scheme.

Besides, we explore an hyper-parameter (τ_q and τ_{cqc}) sensitivity according to the color jitter strength in Figure 6. In general, the difference in performance due to the change of hyper-parameters is insignificant; however, the effect of the color jitter strength is pronounced. As a result, we confirm that SPQ is robust to hyper-parameters, and input data preparation is an important factor.

4.4.2 Visualization

As illustrated in Figure 5, we employ t-SNE [34] to examine the distribution of deep representations of BinGAN, TBH, and our SPQ, where BinGAN and SPQ are trained under the truly unsupervised setting. Nonetheless, our SPQ scatters data samples most distinctly where each color denotes a different class label. Furthermore, we show the actual returned images in Figure 7. Interestingly, not only images of the same category but also images with visually similar contents are retrieved, like a cat appears in the dog retrieval results.

5. Conclusion

In this paper, we have proposed a novel deep self-supervised learning-based fast image retrieval method, Self-supervised Product Quantization (SPQ) network. By employing a product quantization scheme, we built the first end-to-end unsupervised learning framework for image retrieval. We introduced a cross quantized contrastive learning strategy to learn the deep representations and codewords to discriminate the image contents while clustering local patterns at the same time. Despite the absence of any supervised label information, our SPQ yields state-of-the-art retrieval results on three large-scale benchmark datasets. As future research, we expect performance gain by contrasting more views within a batch, which needs a better computing environment. Our code is publicly available at https://github.com/youngkyunJang/SPQ.

6. Acknowledgement

References

[1] Relja Arandjelovic and Andrew Zisserman. All about vlad. In *CVPR*, pages 1578–1585, 2013. 4

[2] Artem Babenko and Victor Lempitsky. Additive quantization for extreme vector compression. In *CVPR*, pages 931–938, 2014. 1, 2

[3] Artem Babenko and Victor Lempitsky. Tree quantization for large-scale similarity search and classification. In *CVPR*, pages 4240–4248, 2015. 1, 2

[4] Yue Cao, Mingsheng Long, Jianmin Wang, Han Zhu, and Qingfu Wen. Deep quantization network for efficient image retrieval. In *AAAI*, 2016. 2

[5] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Philip S Yu. Hashnet: Deep learning to hash by continuation. In *ICCV*, pages 5608–5617, 2017. 2

[6] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *arXiv preprint arXiv:2006.09882*, 2020. 2, 3, 4, 6

[7] Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, pages 380–388, 2002. 1, 2, 5, 6

[8] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *Arxiv*, 2020. 2, 3, 4, 5, 6

[9] Ting Chen, Yizhou Sun, Yue Shi, and Liangjie Hong. On sampling strategies for neural network-based collaborative filtering. In *ACM SIGKDD*, pages 767–776, 2017. 4

[10] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. Nus-wide: a real-world web image database from national university of singapore. In *CIVR*, page 48. ACM, 2009. 5

[11] Bo Dai, Ruiqi Guo, Sanjiv Kumar, Niao He, and Le Song. Stochastic generative hashing. In *ICML*, 2017. 2, 3, 5, 6

[12] et al. E. Riba. A survey on kornia: an open source differentiable computer vision library for pytorch. 2020. 6

[13] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. Optimized product quantization for approximate nearest neighbor search. In *CVPR*, pages 2946–2953, 2013. 1, 2, 5, 6

[14] Kamran Ghasedi Dizaji, Feng Zheng, Najmeh Sadoughi, Yanhua Yang, Cheng Deng, and Heng Huang. Unsupervised deep generative adversarial hashing network. In *CVPR*, pages 3664–3673, 2018. 2, 3, 5, 6

[15] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *PAMI*, 35(12):2916–2929, 2012. 1, 2, 5, 6

[16] Robert M. Gray and David L. Neuhoff. Quantization. *IEEE Transactions on Information Theory*, 44(6):2325–2383, 1998. 1

[17] Kaiming He, Fang Wen, and Jian Sun. K-means hashing: An affinity-preserving quantization method for learning binary compact codes. In *CVPR*, pages 2938–2945, 2013. 1, 2

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 3, 6

[19] Mark J Huiskes and Michael S Lew. The mir flickr retrieval evaluation. In *ICMR*, pages 39–43, 2008. 5

[20] Himalaya Jain, Joaquin Zepeda, Patrick Pérez, and Rémi Gribonval. Subic: A supervised, structured binary code for image search. In *ICCV*, pages 833–842, 2017. 2

[21] Young Kyun Jang and Nam Ik Cho. Generalized product quantization network for semi-supervised image retrieval. In *CVPR*, 2020. 2, 4

[22] Young Kyun Jang, Dong-ju Jeong, Seok Hee Lee, and Nam Ik Cho. Deep clustering and block hashing network for face image retrieval. In *ACCV*, pages 325–339. Springer, 2018. 2

[23] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *PAMI*, 33(1):117–128, 2010. 1, 2, 5, 6

[24] Yannis Kalantidis and Yannis Avrithis. Locally optimized product quantization for approximate nearest neighbor search. In *CVPR*, pages 2321–2328, 2014. 1, 2, 5, 6

[25] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2015. 6

[26] Benjamin Klein and Lior Wolf. End-to-end supervised product quantization for image search and retrieval. In *CVPR*, pages 5041–5050, 2019. 2

[27] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009. 5

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, pages 1097–1105, 2012. 3, 5

[29] Qi Li, Zhenan Sun, Ran He, and Tieniu Tan. Deep supervised discrete hashing. In *NeurIPS*, pages 2482–2491, 2017. 2

[30] Kevin Lin, Jiwen Lu, Chu-Song Chen, and Jie Zhou. Learning compact binary descriptors with unsupervised deep neural networks. In *CVPR*, pages 1183–1192, 2016. 2, 3, 5, 6

[31] Bin Liu, Yue Cao, Mingsheng Long, Jianmin Wang, and Jingdong Wang. Deep triplet quantization. *ACMMM*, 2018. 2

[32] Wei Liu, Cun Mu, Sanjiv Kumar, and Shih-Fu Chang. Discrete graph hashing. In *NeurIPS*, pages 3419–3427, 2014. 1, 2

[33] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 6

[34] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008. 8

[35] Stanislav Morozov and Artem Babenko. Unsupervised neural quantization for compressed-domain similarity search. In *ICCV*, pages 3036–3045, 2019. 2, 3

[36] Yuming Shen, Li Liu, and Ling Shao. Unsupervised binary representation learning with deep variational networks. *IJCV*, 127(11-12):1614–1628, 2019. 2, 3, 5, 6

[37] Yuming Shen, Jie Qin, Jiaxin Chen, Mengyang Yu, Li Liu, Fan Zhu, Fumin Shen, and Ling Shao. Auto-encoding twin-bottleneck hashing. In *CVPR*, pages 2818–2827, 2020. 2, 3, 5, 6, 7, 8

[38] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015. 3, 5

[39] Jingkuan Song. Binary generative adversarial networks for image retrieval. In *AAAI*, 2017. 2, 3, 5, 6

[40] Aravind Srinivas, Michael Laskin, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. *arXiv preprint arXiv:2004.04136*, 2020. 2, 3, 4

[41] Shupeng Su, Chao Zhang, Kai Han, and Yonghong Tian. Greedy hash: Towards fast optimization for accurate hash coding in cnn. In *NeurIPS*, pages 798–807, 2018. 2, 5, 6

[42] Jingdong Wang, Ting Zhang, Nicu Sebe, Heng Tao Shen, et al. A survey on learning to hash. *PAMI*, 40(4):769–790, 2017. 1, 2

[43] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In *NeurIPS*, pages 1753–1760, 2009. 1, 2, 5, 6

[44] Rongkai Xia, Yan Pan, Hanjiang Lai, Cong Liu, and Shuicheng Yan. Supervised hashing for image retrieval via image representation learning. In *AAAI*, 2014. 2

[45] Erkun Yang, Tongliang Liu, Cheng Deng, Wei Liu, and Dacheng Tao. Distillhash: Unsupervised deep hashing by distilling data pairs. In *CVPR*, pages 2946–2955, 2019. 2, 5, 6

[46] Tan Yu, Jingjing Meng, Chen Fang, Hailin Jin, and Junsong Yuan. Product quantization network for fast visual search. *IJCV*, pages 1–19, 2020. 2, 4

[47] 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 *CVPR*, pages 3083–3092, 2020. 2

[48] Ting Zhang, Chao Du, and Jingdong Wang. Composite quantization for approximate nearest neighbor search. In *ICML*, volume 2, page 3, 2014. 1, 2

[49] Ting Zhang, Guo-Jun Qi, Jinhui Tang, and Jingdong Wang. Sparse composite quantization. In *CVPR*, pages 4548–4556, 2015. 1, 2

[50] Maciej Zieba, Piotr Semberecki, Tarek El-Gaaly, and Tomasz Trzcinski. Bingan: Learning compact binary descriptors with a regularized gan. In *NeurIPS*, pages 3608–3618, 2018. 2, 3, 5, 6, 8