Product Quantization Network for Fast Image Retrieval

Tan Yu[1], Junsong Yuan[2], Chen Fang[3], and Hailin Jin[3]

[1] Nanyang Technological University, Singapore
tyu008@e.ntu.edu.sg
[2] State University of New York at Buffalo
jsyuan@buffalo.edu
[3] Adobe Research
{cfang, hljin}@adobe.com

Abstract. Product quantization has been widely used in fast image retrieval due to its effectiveness of coding high-dimensional visual features. By extending the hard assignment to soft assignment, we make it feasible to incorporate the product quantization as a layer of a convolutional neural network and propose our product quantization network. Meanwhile, we come up with a novel asymmetric triplet loss, which effectively boosts the retrieval accuracy of the proposed product quantization network based on asymmetric similarity. Through the proposed product quantization network, we can obtain a discriminative and compact image representation in an end-to-end manner, which further enables a fast and accurate image retrieval. Comprehensive experiments conducted on public benchmark datasets demonstrate the state-of-the-art performance of the proposed product quantization network.

1 Introduction

Image retrieval has been a fundamental research topic in computer vision. Given a query image, it aims to find the relevant images from a database. Precision and efficiency are two key aspects for a retrieval system. These two key aspects also drives the image retrieval research to progress in two directions.

One direction is to design or learn a more effective image representation for a higher retrieval precision [31, 30, 18, 3, 2, 28, 11, 40, 4, 5]. Good image representation maintains a large distance between irrelevant images in feature space and keeps a close distance between relevant images. Traditional image retrieval systems generated image representation by aggregating hand-craft local features like SIFT and the research focuses on designing a more effective aggregation method [31, 30, 18]. With the progress of deep learning, the convolutional neural network provides an effective image representation[3, 2, 28, 38, 41], which is trained by the semantic information and is robust to low-level image transformations.

On the other hand, to achieve a satisfactory efficiency in image retrieval, especially when dealing with a large-scale dataset, a compact image representation is necessary. Generically speaking, there are two types of schemes to gain

a compact image representation, hashing and quantization. Hashing maps the real-value vectors into binary codes, which enables a faster distance computation and lower memory cost. One of most widely used hashing method is locality sensitivity hashing (LSH) [7]. Nevertheless, LSH is data-independent, which ignores the data distribution and is sub-optimal to a specific dataset. To further improve the performance, some hashing methods [32, 36, 10] learn the projection from the data, which caters better to a specific dataset and achieves higher retrieval precision. Traditional hashing methods are based on off-the-shelf visual features. They optimize the feature extraction and Hamming embedding independently. More recently, inspired by the progress of deep learning, some deep hashing methods [37, 22, 25, 23] are proposed, which simultaneously conduct the feature learning and compression through a unified network.

Nevertheless, hashing methods are only able to produce a few distinct distances, limiting its capability of describing the distance between data points. In parallel to hashing methods, another widely used data compression method in image retrieval is product quantization. It represents each feature vector by a Cartesian product of several codewords. Thanks to the asymmetric distance calculation mechanism, it enables a more accurate distance calculation than hashing methods using the same code length. The product quantization (PQ) [17] and its optimized versions like OPQ [9], CKmeans [29], APQ [1] and CQ [43, 12] are originally designed for an unsupervised scenario where no labeling data are provided. SQ [35] extends product quantization to a supervised scenario. However, SQ is based on the hand-crafted features or CNN features from the pretrained model, therefore it might not be optimal with respect to the a specific dataset.

To simultaneously optimize the feature learning and quantization, Cao *et al* [6] propose a deep quantization network (DQN) which can be trained in an end-to-end manner. It optimizes a weighted sum of similarity-preserve loss and product quantization loss. It iteratively updates codewords and other parameters of a neural network. Therefore, in each iteration, the codewords are directly updated by k-means whereas the label information is ignored. Recently, Klein *et al.* [20] propose a deep product quantization (DPQ). They learn a cascade of two fully-connected layers followed by a softmax layer to determine a soft codeword assignment. It is different from original product quantization, the codeword assignment is no longer determined by distance between the original feature and codewords. Nevertheless, the additional parameters introduced in the cascade of fully-connected layers make DPQ more prone to over-fitting.

In this paper, we also attempt to incorporate the product quantization in a neural network and train it in an end-to-end manner. We propose a soft product quantization layer which is differentiable and the original product quantization is a special case of the proposed soft product quantization when $\alpha \to +\infty$. Different from DPQ, we no longer need fully-connected layers to obtain the codebook assignment, instead, in our method, the codeword assignement is determined by the similarity between the original feature and the codewords. Therefore, we significantly reduce the number of parameters to be trained, making our PQN more immune to over-fitting compared with DPQ. Meanwhile, inspired by the

success of the triplet loss in metric learning and the triumph of the asymmetric similarity measurement in feature compression, we propose a novel asymmetric triplet loss to directly optimize the asymmetric similarity measurement in an end-to-end manner. In summary, the contribution of our work is three-fold:

- We introduce a novel soft product quantization layer, which is a generalized version of the original product quantization. It is differentiable and thus brings an end-to-end training of the product quantization network.
- We propose a novel asymmetric triplet loss, which directly optimizes the asymmetric distance brought based on product quantization. It enables a more effective training of the convolutional neural network.
- Due to its simplicity, effectiveness and efficiency, we provide the image retrieval community a strong baseline. Some more sophisticated image retrieval methods can be further built upon the proposed framework.

2 Related Work

Hashing [7, 32, 36, 10, 37, 22, 25, 23, 15, 14, 13, 39] aims to map a feature vector into a short code consisting of a sequence of bits, which enables a fast distance computation mechanism as well as a light memory cost. Traditional hashing methods like locality sensitivity hashing (LSH) [7], spetral hashing (SH) [36] and iterative quantization (ITQ) [10] first obtain real-value image features and then compress the features into binary codes. They conduct the representation learning and the feature compression separately and the mutual influence between them is ignored. Recently, motivated by the success of deep learning, some works [37, 22, 25, 23] propose deep hashing methods by incorporating hashing as a layer into a deep neural network. The end-to-end training mechanism of deep hashing simultaneously optimizes the representation learning and feature compression, achieving better performance than the traditional hashing methods.

Since the hashing methods are only able to produce a few distinct distances, it has limited capability of describing the distance between data points. Parallelly, another scheme termed product quantization (PQ) [17] decomposes the space into a Cartesian product of subspaces and quantizes each subspace individually. Some following works [9, 1, 43] further optimize the product quantization through reducing the distortion errors and achieve higher retrieval precision. Note that production quantization and its optimized versions such as OPQ [9], AQ [1] and CQ [43] are originally designed for an unsupervised scenario where no label information is provided.

Wang *et al* [35] propose supervised quantization (SQ) by exploiting the label information. Nevertheless, SQ conducts feature extraction and quantization individually, whereas the interaction between these two steps are ignored. To simultaneously learn image representation and product quantization, deep quantization network (DQN) [6] adds a fully connected bottleneck layer in the convolutional network. It optimizes a combined loss consisting of a similarity-preserving loss and a product quantization loss. Nevertheless, the codebook in DPQ is trained

through k-means clustering and thus the supervised information is ignored. Recently, deep product quantization (DPQ) [20] is proposed where the codebook as well as the parameters are learned in an end-to-end manner. Different from original product quantization which determines the codeword assignment according to the distance between the original feature and codewords, DPQ determines the codeword assignment through a fully-connected layer whose parameters are learned from data. Nevertheless, the additional parameters in the cascade of fully-connected layers will make the network more prone to over-fitting.

Our work is also an attempt of incorporating the product quantization in a neural network. We propose a soft product quantization layer and build our product quantization network (PQN), which can be trained in an end-to-end manner. Different from DPQ, our PQN determines the codeword assignment according to the similarity between the feature for coding and codewords, which can be seen as a soft extension of original product quantization. Unlike DPQ, we do not need additional fully-connected layers to determine the codeword assignment and the parameters in our soft product quantization layer are only the codewords. Therefore, the number of parameters in our quantization layer is considerably less than that of DPQ, which mitigates the over-fitting. As shown in experiments, our PQN consistently outperforms DPQ by a large margin.

3 Product Quantization Network

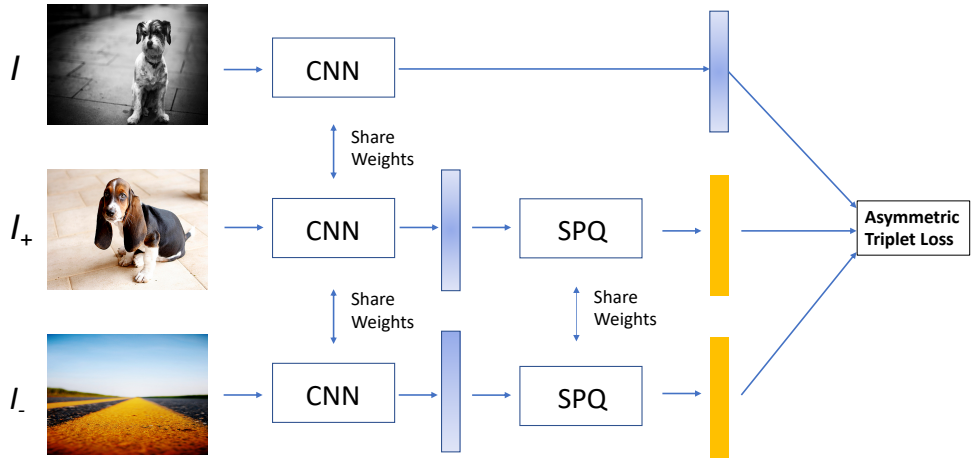


Fig. 1. The overview of the proposed product quantization network, where CNN represents the convolutional neural network and SPQ represents the proposed soft product quantization layer. The asymmetric triplet loss takes as input a triplet consisting of the CNN feature of an anchor image (I), the SPQ feature of a positive sample (I_+) and the SPQ feature of a negative sample (I_-).

3.1 From Hard Quantization to Soft Quantization

Let us denote by $\mathbf{x} \in \mathbb{R}^d$ the feature of an image I, we divide the feature $\mathbf{x}$ into M subvectors $[\mathbf{x}_1, \cdots, \mathbf{x}_m, \cdots, \mathbf{x}_M]$ in the feature space where $\mathbf{x}_m \in \mathbb{R}^{d/M}$ is a subvector. The product quantization further approximates $\mathbf{x}$ by

$$\mathbf{q} = [q_1(\mathbf{x}_1), \cdots, q_m(\mathbf{x}_m), \cdots, q_M(\mathbf{x}_M)], \tag{1}$$

where $q_m(\cdot)$ is the quantizer for $\mathbf{x}_m$ defined as

$$q_m(\mathbf{x}_m) = \sum_k \mathbb{1}(k = k^*)\mathbf{c}_{mk}, \tag{2}$$

where $k^* = \underset{k}{\operatorname{argmin}} \|\mathbf{c}_{mk} - \mathbf{x}_m\|_2$, $\mathbb{1}(\cdot)$ is the indicator function and $\mathbf{c}_{mk}$ is the k-th codeword from the m-th codebook. The hard assignment makes it infeasible to derive its derivative and thus it can not be incorporated in a neural network. This embarrassment motivates us to replace the hard assignment $\mathbb{1}(k = k^*)$ by the soft assignment $e^{-\alpha\|\mathbf{x}_m - \mathbf{c}_{mk}\|_2^2} / \sum_{k'} e^{-\alpha\|\mathbf{x}_m - \mathbf{c}_{mk'}\|_2^2}$ and obtain

$$\mathbf{s} = [s_1(\mathbf{x}_1), \cdots, s_m(\mathbf{x}_m), \cdots, s_M(\mathbf{x}_M)], \tag{3}$$

where $s_m(\cdot)$ is the soft quantizer for m-th subvector defined as

$$s_m(\mathbf{x}_m) = \sum_k \frac{e^{-\alpha\|\mathbf{x}_m - \mathbf{c}_{mk}\|_2^2}\mathbf{c}_{mk}}{\sum_{k'} e^{-\alpha\|\mathbf{x}_m - \mathbf{c}_{mk'}\|_2^2}}. \tag{4}$$

It is not difficult to observe that

$$\mathbb{1}(k = k^*) = \lim_{\alpha \to +\infty} \frac{e^{-\alpha\|\mathbf{x}_m - \mathbf{c}_{mk}\|_2^2}}{\sum_{k'} e^{-\alpha\|\mathbf{x}_m - \mathbf{c}_{mk'}\|_2^2}} \tag{5}$$

Therefore, when $\alpha \to +\infty$, the soft quantizer $s_m(\mathbf{x}_m)$ will be equivalent to the hard quantizer $q_m(\mathbf{x}_m)$. Since the soft quantization operation is differentiable and thus it can be readily incorporated into a network as a layer.

3.2 Soft Product Quantization Layer

Before we conduct soft product quantization in the network, we first pre-process the original feature $\mathbf{x} = [\mathbf{x}_1, \cdots, \mathbf{x}_m, \cdots, \mathbf{x}_M]$ through intra-normalization and conduct ℓ_2-normalization on codewords $\{\mathbf{c}_{mk}\}_{m=1,k=1}^{M,K}$:

$$\mathbf{x}_m \leftarrow \mathbf{x}_m / \|\mathbf{x}_m\|_2 \tag{6}$$

$$\mathbf{c}_{mk} \leftarrow \mathbf{c}_{mk} / \|\mathbf{c}_{mk}\|_2 \tag{7}$$

The pre-processing step is motivated by two reasons: 1) intra-normalization and ℓ_2-normalization can balance the contribution of each sub-vector and each codeword; 2) it simplifies the gradient computation.

Forward pass. After intra-normalization on original features and ℓ_2-normalization on the codewords, we can obtain $\|\mathbf{x}_m - \mathbf{c}_{mk}\|_2^2 = 2 - 2\langle \mathbf{x}_m, \mathbf{c}_{mk} \rangle$, where $\langle \cdot, \cdot \rangle$ denotes the inner product between two vectors. Based on the above property, we can rewrite Eq. (4) into:

$$s_m(\mathbf{x}_m) = \sum_k \frac{e^{2\alpha \langle \mathbf{x}_m, \mathbf{c}_{mk} \rangle} \mathbf{c}_{mk}}{\sum_{k'} e^{2\alpha \langle \mathbf{x}_m, \mathbf{c}_{mk'} \rangle}}. \tag{8}$$

Backward pass. To elaborate the backward pass of the soft quantization layer, we introduce an immediate variable a_{mk} defined as

$$a_{mk} = \frac{e^{2\alpha \langle \mathbf{x}_m, \mathbf{c}_{mk} \rangle}}{\sum_{k'} e^{2\alpha \langle \mathbf{x}_m, \mathbf{c}_{mk'} \rangle}}. \tag{9}$$

Based on the above definition, Eq. (8) will be converted into

$$s_m(\mathbf{x}_m) = \sum_k a_{mk} \mathbf{c}_{mk}. \tag{10}$$

Through chain rule, we can obtain the derivative of loss with respect to $\mathbf{c}_{mk}$ by

$$\frac{\partial L}{\partial \mathbf{c}_{mk}} = a_{mk} \frac{\partial L}{\partial s_m(\mathbf{x}_m)} + \sum_{k'} \frac{\partial a_{mk'}}{\partial \mathbf{c}_{mk}} \left(\frac{\partial s_m(\mathbf{x}_m)}{\partial a_{mk'}} \right)^\top \frac{\partial L}{\partial s_m(\mathbf{x}_m)}, \tag{11}$$

where

$$\frac{\partial s_m(\mathbf{x}_m)}{\partial a_{mk'}} = \mathbf{c}_{mk'}, \tag{12}$$

and

$$\frac{\partial a_{mk'}}{\partial \mathbf{c}_{mk}} = \begin{cases} \dfrac{-e^{2\alpha \langle \mathbf{x}_m, \mathbf{c}_{mk'} \rangle} e^{2\alpha \langle \mathbf{x}_m, \mathbf{c}_{mk} \rangle} 2\alpha \mathbf{x}_m}{(\sum_{k''} e^{2\alpha \langle \mathbf{x}_m, \mathbf{c}_{mk''} \rangle})^2}, & k \neq k' \\ \dfrac{\sum_{k' \neq k} e^{2\alpha \langle \mathbf{x}_m, \mathbf{c}_{mk'} \rangle} e^{2\alpha \langle \mathbf{x}_m, \mathbf{c}_{mk} \rangle} 2\alpha \mathbf{x}_m}{(\sum_{k''} e^{2\alpha \langle \mathbf{x}_m, \mathbf{c}_{mk''} \rangle})^2}, & k = k' \end{cases} \tag{13}$$

By plugging Eq. (12) and Eq. (13) into Eq. (11), we can obtain $\frac{\partial L}{\partial \mathbf{c}_{mk}}$.

3.3 Initialization

We initialize the parameters of convolutional layers by fine-tuning a standard convolutional neural network without quantization, *e.g.*, Alexnet, on the specific dataset. Note that, we add an intra-normalization layer to fine-tune the network to make it compatible with our deep product quantization network. After the initialization of convolutional layers, we extract the features from the fine-tuned network and conduct k-means followed by ℓ_2-normalization to obtain the initialized codewords $\{\mathbf{c}_{mk}\}_{k=1,m=1}^{K,M}$ in the soft product quantization layer.

3.4 Asymmetric Triplet Loss

We propose a novel asymmetric triplet loss to optimize the parameters of the network. We define (I, I_+, I_-) as a training triplet, where I_- and I_+ represent a relevant image and an irrelevant image with respect to the anchor image I. We denote by $\mathbf{x}_I$ as the feature of I before soft product quantization and denote by $\mathbf{s}_{I_+}$ and $\mathbf{s}_{I_-}$ the features of I_+ and I_- after soft product quantization. We define asymmetric similarity between I and $I+$ as $\langle \mathbf{x}_I, \mathbf{s}_{I_+} \rangle$, where $\langle \cdot, \cdot \rangle$ denotes the inner-product operation. The proposed asymmetric triplet loss is defined as

$$l = \langle \mathbf{x}_I, \mathbf{s}_{I_-} \rangle - \langle \mathbf{x}_I, \mathbf{s}_{I_+} \rangle. \tag{14}$$

Intuitively, it aims to increase the asymmetric similarity between the pairs of relevant images and decrease that of pairs consisting of irrelevant images. It is a natural extension of original triplet loss on the condition of asymmetric distance. The difference is that, a training triplet used in original triplet loss consists of three features of the same type, whereas a training triplet used in the proposed asymmetric triplet loss consists of one feature without quantization and two features after quantization. In fact, our experiments show that a better performance is achieved by processing above loss through sigmoid function and a revised loss function is defined as Eq. (15). The better performance might be contributed by the fact that the sigmoid function can normalize the original loss so that the training will not be biased by some samples causing huge loss.

$$l = \frac{1}{1 + e^{\langle \mathbf{x}_I, \mathbf{s}_{I_+} \rangle - \langle \mathbf{x}_I, \mathbf{s}_{I_-} \rangle}}. \tag{15}$$

3.5 Encode and retrieval.

After training the proposed product quantization network, the reference images in the database will be encoded by hard product quantization. We define the layer before the soft product quantization layer as embedding layer. Given a reference image I of the database, we obtain its output from embedding layer $\mathbf{x} = [\mathbf{x}_1, \cdots, \mathbf{x}_m, \cdots, \mathbf{x}_M]$ and further obtain its product quantization code $\mathbf{b} = [b_1, \cdots, b_m, \cdots, b_M]$ where b_m is computed by

$$b_m = \underset{k}{\operatorname{argmax}} \langle \mathbf{x}_m, \mathbf{c}_{mk} \rangle, \tag{16}$$

where $\{\mathbf{c}_{mk}\}_{m=1,k=1}^{M,K}$ are codewords learned from our product quantization network. In the retrieval phase, we obtain the query feature from the embedding layer $\mathbf{q} = [\mathbf{q}_1, \cdots, \mathbf{q}_m, \cdots, \mathbf{q}_M]$. The relevance between the query image and a reference image represented by its product quantization code $\mathbf{b} = [b_1, \cdots, b_m, \cdots, b_M]$ is computed by the asymmetric similarity $s(\mathbf{q}, \mathbf{b})$ defined as

$$s(\mathbf{q}, \mathbf{b}) = \sum_{m=1}^{M} \langle \mathbf{q}_m, \mathbf{c}_{mb_m} \rangle. \tag{17}$$

Since $\langle \mathbf{q}_m, \mathbf{c}_{mb_m} \rangle$ is computed only once for all the reference images in the database and thus obtaining $s(\mathbf{q}, \mathbf{b})$ only requires to sum up the pre-computed similarity scores in the look-up table, considerably speeding up the image retrieval process. Meanwhile, storing the product quantization code $\mathbf{b}$ only requires $M \log_2 K$ bits, which considerably reduces the memory cost.

3.6 Relation to existing methods.

DQN [6] is the first attempt of incorporating product quantization in the neural network. It alternatively optimizes codewords and other parameters of the network. It is worth noting that when updating codewords, it only minimizes the quantizaition errors through k-means. Therefore, when learning codewords, the supervision information is ignored and the solution might be sub-optimal.

SUBIC [16] integrates the one-hot block encoding layer in the deep neural network. It represents each image by a product of one-hot blocks, following the spirit of product quantization. Nevertheless, the sparse property limits its representation capability, making it perform not as well as ours.

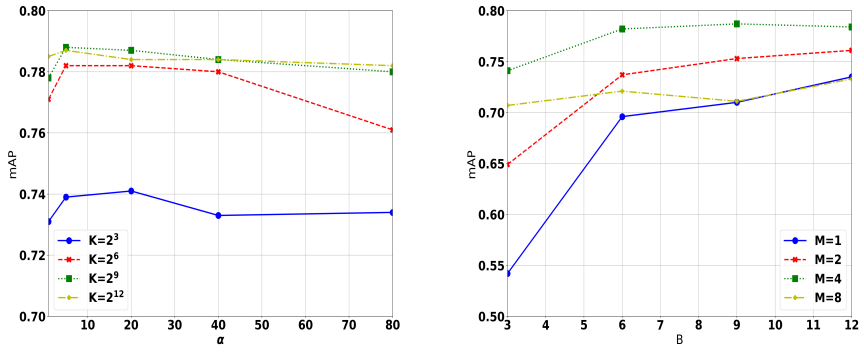
DPQ [20] is another attempt of incorporating the product quantization into the neural network. It determines the codeword assignment through a cascade of two fully-connected layers. In contrast, our method determines the codeword assignment according to the similarity between original feature and the codewords. Note that, the additional parameters from these two fully-connected layers in DPQ no only increase the computation complexity in training the neural network but also are more prone to over-fitting. Our experiments show that our proposed PQN considerably outperforms DPQ.

4 Experiments

We evaluate the performance of our PQN on two public benchmark datasets, CIFAR-10 and NUS-WIDE. CIFAR-10 [21] is a dataset containing $60,000$ color images in 10 classes, and each class has $6,000$ images in size 32×32. Different from CIFAR-10, NUS-WIDE [8] is a dataset for evaluating multi-class classification, in which one sample is assigned to one or multiple labels. We follow the settings in [22, 6] and use the subset of $195,834$ images that are associated with the 21 most frequent concepts, where each concept consists of at least $5,000$ images. We resize all images into 256×256.

On the CIFAR-10 dataset, the performance reported by different baselines are based on different base convolutional neural networks, making it unfair to directly compare their reported retrieval accuracy. To make a fair comparison, we evaluate our method based on two types of convolutional neural networks. The first convolutional neural network we use is 3CNet which is also used by SUBIC

[16] and DQN [20]. 3CNet is proposed in [25], which consists of $L = 3$ convolutional layers with 32, 32 and 64 filters of size 5×5 respectively, followed by a fully connected layer with $d = 500$ nodes. The second convolutional neural network we choose is AlexNet. It is worth noting that the baselines we compare may apply different models. For example, DQN [6] adopts AlexNet whereas other work [34, 23] adopt VGG-F model. These two models are similar in the architecture. To be specific, both the CNN-F and AlexNet consist of five convolutional layers and two fully connected layers. As shown in [19], the CNN-F generally performs better than Alexnet in image retrieval, therefore, the better performance of ours based on AlexNet than existing state-of-art methods based on CNN-F is not owing to better base network. In other words, our method can achieve better performance even with an inferior base network. On the NUS-WIDE dataset, we also adopt AlexNet as our base model. On both datasets, we report the performance of the proposed method through mAP, which is a standard metric in evaluating the performance of retrieval algorithms.



(a) Influence of α (b) Influence of M and K

Fig. 2. The influence of parameters on CIFAR-10 dataset using 3CNet.

4.1 CIFAR-10 using 3CNet

Following the experimental setup in SUBIC [16] and DPQ [20], the training is conducted on 50K image training set. The test set is split into 9K database images and 1K query images (100 per class).

Influence of M and K. In this section, we evaluate the influence of the number of subvectors M and the number of codewords per sub-codebook K on the retrieval precision of the proposed PQN. We vary M among $\{1, 2, 4, 8\}$, and vary $B = log_2 K$ among $\{3, 6, 9, 12\}$. As shown in Fig. 2(a), the proposed method

achieves the best performance when $M = 4$. By default, we set $M = 4$ on CIFAR-10 Dataset. Note that when $M = 1$ and $M = 2$, the performance of the proposed PQN increases as B increases. This is expected since the larger B can partition the feature space into finer cells. Nevertheless, when $M = 4$, the performance drops when B increases from 9 to 12. Meanwhile, when $M = 8$, there is also a performance drop when K increases from 6 to 9. The worse performance might be caused by over-fitting when both M and K are large.

Influence of α. α controls the quantization softness of the soft product quantization layer. We evaluate the performance of our method when α varies. We test the influence of α when $M = 4$ and K varies among $\{2^3, 2^6, 2^9, 2^{12}\}$. As shown in Fig. 2(b), the performance of the proposed PQN is relatively stable when α increases from 1 to 80. Note that, when $\alpha = 1$, the performance is slightly worse than that when $\alpha = 5$. The worse performance is due to the fact a small α will make the quantization too soft and thus the soft quantization in training phase differs too much from the hard quantization in the testing phase. Meanwhile, we also observe a performance drop when α increases from 20 to 40. This drops might be caused by the fact that a huge α tends to push the input of soft-max function to the saturation region and lead to gradient vanishing.

Comparison with unsupervised PQ/LSQ. We compare with unsupervised PQ and LSQ [27] based on fine-tuned features trained through triplet loss. As shown in Table 1, ours considerably outperforms both TL+PQ and TL+LSQ. Meanwhile, we also show the performance of original features trained through triplet loss without quantization (TL+Full) in Table 1. The performance of ours is even better than that of features without quantization, this is owing to the regularization imposed by quantization, which suppresses over-fitting.

method	4bits	8bits	16bits	24bits	32bits
TL+Full	0.779				
TL+PQ	0.503	0.621	0.741	0.773	0.780
TL+LSQ	0.511	0.720	0.752	0.753	0.763
PQN(Ours)	**0.574**	**0.729**	**0.778**	**0.782**	**0.786**

Table 1. Comparisons with PQ and LSQ.

Effectiveness of asymmetric triplet loss. Meanwhile, in order to show the effectiveness of the proposed asymmetric triplet loss, we compare with two alternatives, cross-entropy loss (CEL) and triplet loss (TL). To make a fair comparison, we only change the loss function and keep the other parts of the network unchanged. As shown in Fig. 3, the proposed assymetric loss consistently outperforms the cross-entropy loss and triplet loss when L varies among $\{12, 24, 36, 48\}$.

For instance, when $L = 36$, our ATL achieves a 0.787 mAP whereas TL only achieves a 0.778 mAP and CEL only achieves a 0.768 mAP.

Compare with state-of-the-art methods. We compare our method with two state-of-the-art methods (SUBIC and DPQ), which adopt the same 3CNet as well as the same experimental settings. We change bit length L among $\{12, 24, 36, 48\}$. We set $M = 4$ and $K = 8, 64, 512, 4096$, respectively. Since SUBIC adopts cross-entropy loss, it is unfair to directly compare it with ours using asymmetric triplet loss. Therefore, we report the performance of our PQN based on the cross-entropy loss (CEL) as well as the proposed asymmetric triplet loss (ATL). As shown in Table 2, our method based on both CEL and ATL significantly outperform the existing state-of-the-art methods including SUBIC and DPQ. For instance, when $L = 24$, ours achieves a 0.771 mAP based on the cross-entropy loss and a 0.782 mAP using the proposed asymmetric triplet loss whereas SUBIC only achieves a 0.672 mAP and DPQ only achieves a 0.692 mAP.

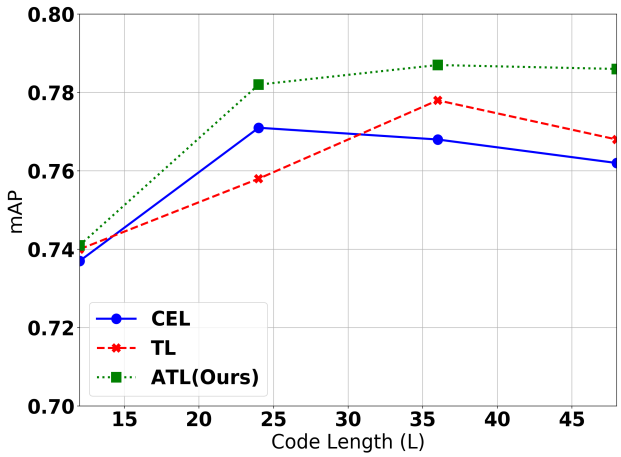


Fig. 3. Comparisons of our asymmetric triple loss (ATL) with cross-entropy loss (CEL) and triplet loss (TL).

4.2 CIFAR-10 using AlexNet

Following the experimental settings in [34, 23], we randomly sample 1000 images per class (10000 images in total) as the testing query images, and the remaining 50000 images are used as the training set as well as reference images in the database. We set $M = 4$ and vary K among $\{2^4, 2^6, 2^9, 2^{12}\}$, and thus the code length L varies among $\{16, 24, 36, 48\}$.

Method	12 bits	24 bits	36 bits	48 bits
SUBIC [16]	0.635	0.672	0.682	0.686
DPQ [20]	0.673	0.692	0.695	0.693
Ours+CEL	**0.737**	**0.771**	**0.768**	**0.762**
Ours+ATL	**0.741**	**0.782**	**0.787**	**0.786**

Table 2. mAP comparisons with state-of-the-art methods using 3CNet.

Comparions with state-of-the-art methods. As shown in Table 3, ours consistently outperforms the existing state-of-the-art methods, especially when the bit length is small. For instance, when the bit length is 16, our method based on asymmetric triplet loss (Ours+ATL) achieves a 0.947 mAP whereas the second best method, DSDH, only achieves a 0.935 mAP.

Method	16 bits	24 bits	36 bits	48 bits
DRSCH [42]	0.615	0.622	0.629	0.631
DSCH [42]	0.609	0.613	0.617	0.686
DSRH [45]	0.608	0.611	0.617	0.618
VDSH [44]	0.845	0.848	0.844	0.845
DPSH [24]	0.903	0.885	0.915	0.911
DTSH [34]	0.915	0.923	0.925	0.926
DSDH [23]	0.935	0.940	0.939	0.939
Ours + CE	0.939	0.941	0.941	0.940
Ours + ATL	**0.947**	**0.947**	**0.946**	**0.947**

Table 3. mAP comparisons with existing state-of-the-art methods using AlexNet base model on the CIFAR10 dataset.

Extremely short code evaluation. As shown in Table 3, the mAP achieved by our method does not drop when the bit length decreases from 48 to 16. In contrast, the performance of other methods in Table 3 all turn worse due to decrease of the bit length. To fully exploit the potential of the proposed product quantization network on the CIAFR-10 dataset, we evaluate it by setting the code length L extremely small. We vary M among 1, 2 and 4, and meanwhile vary the code length (bit number) L within $\{4, 6, 8, 10, 12\}$. As shown in Fig. 4, when code length is extremely small, *e.g.*, $L = 4$, the performance of PQN when $M = 1$ significantly outperforms that when $M = 2, 4$. Meanwhile, when $M = 1$, there is not significant performance drop when L decreases from 12 to 4. Note that, when $M = 1$, the proposed PQN achieves a 0.945 mAP when using only 4 bits per code. It considerably outperforms the existing state-of-art method DSDH [23] which only achieves 0.935 mAP using 16 bits.

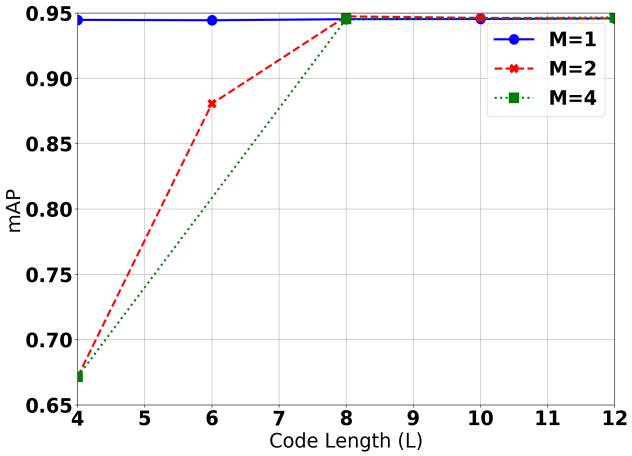


Fig. 4. Evaluation on the extremely short code.

4.3 NUS-WIDE

Following the experiment setup in [34, 23], we randomly sample 100 images per class (2100 images in total) as the test query set and the remaining images are used as database images. 500 database images per label are randomly sampled as training images. The mAP is calculated based on the top 5000 returned neighbors. Due to multi-label settings, the cross-entropy loss used in SUBIC [16] and the softmax loss in DPQ [20] are no longer feasible, which explains neither SUBIC [16] nor DPQ [20] conducts the experiments on NUS-WIDE dataset. Inspired by the success of label embedding proposed in [23], we also adopt a combined loss, which is a weighed sum of our asymmetric triplet loss and a mean square loss defined as

$$l = \frac{1}{1 + e^{\langle \mathbf{x}_I, \mathbf{s}_{I_+} \rangle - \langle \mathbf{x}_I, \mathbf{s}_{I_-} \rangle}} + \beta \|\mathbf{W}\mathbf{s}_I - \mathbf{y}_I\|_2^2, \tag{18}$$

where $\mathbf{W}$ is the parameter in an additional FC layer after the soft product quantization layer and $\mathbf{y}_I$ is the label of the sample I. We set $\beta = 10$ by default.

Comparisons with existing state-of-the-art methods. We compare our method with two types of baselines. The first type extracts the features from CNN and then convert the extracted features into binary codes. We directly copy the reported results in [23] which conducts experiments on several traditional hashing methods such as SH [32], ITQ [10], KSH [26], SDH [33], *etc.* The baselines of the second type are deep hashing/quantization methods, where the binary codes are learned in an end-to-end manner. We compare several methods of the second type such as DQN [6], DPSH [24], DTSH [34], DSDH [23], *etc.* As shown in Table 4, the proposed PQN consistently outperforms these two types

Method	12 bits	24 bits	36 bits	48 bits
SH + CNN [23]	0.621	0.616	0.615	0.612
ITQ + CNN [23]	0.719	0.739	0.747	0.756
LFH + CNN [23]	0.695	0.734	0.739	0.759
KSH + CNN [23]	0.768	0.786	0.790	0.799
SDH+ CNN [23]	0.780	0.804	0.815	0.824
FASTH+CNN [23]	0.779	0.807	0.816	0.825
CNNH [37]	0.611	0.618	0.625	0.608
NINH [22]	0.674	0.697	0.713	0.715
DHN [46]	0.708	0.735	0.748	0.758
DQN [6]	0.768	0.776	0.783	0.792
DPSH [24]	0.752	0.790	0.794	0.812
DTSH [34]	0.773	0.808	0.812	0.824
DSDH [23]	0.776	0.808	0.820	0.829
Ours	**0.795**	**0.819**	**0.823**	**0.830**

Table 4. mAP comparisons with existing state-of-the-art methods using AlexNet base model on the NUS-WIDE dataset. The mAP is based on top 5000 nearest neighbors.

of baselines when code length L varies among $\{12, 24, 36, 48\}$. The advantage of our PQN over other methods is more obvious when the code length L is short. For instance, when $L = 12$, our PQN achieves a 0.795 mAP whereas the second best method, FASTH+CNN [23] only achieves a 0.779 mAP.

5 Conclusion

In this paper, by incorporating product quantization in the neural network, we propose product quantization nework (PQN) to learn a discriminative and compact image representation in an end-to-end manner. Meanwhile, we propose a novel asymmetric triplet loss, which directly optimizes the image retrieval based on asymmetric distance to train our network more effectively. Systematic experiments conducted on benchmark datasets demonstrate the state-of-the-art performance of the proposed PQN.

Acknowledgement This work is supported in part by start-up grants of University at Buffalo, Computer Science and Engineering Department. This work is supported in part by Singapore Ministry of Education Academic Research Fund Tier 2 MOE2015-T2-2-114. This research was carried out at the ROSE Lab of Nanyang Technological University, Singapore. The ROSE Lab is supported by the National Research Foundation, Prime Ministers Office, Singapore. We gratefully acknowledge the support of NVAITC (NVIDIA AI Technology Centre) for their donation for our research.

References

1. Babenko, A., Lempitsky, V.: Additive quantization for extreme vector compression. In: CVPR. pp. 931–938 (2014)
2. Babenko, A., Lempitsky, V.: Aggregating local deep features for image retrieval. In: ICCV. pp. 1269–1277 (2015)
3. Babenko, A., Slesarev, A., Chigorin, A., Lempitsky, V.: Neural codes for image retrieval. In: ECCV. pp. 584–599. Springer (2014)
4. Bai, S., Bai, X., Tian, Q., Latecki, L.J.: Regularized diffusion process on bidirectional context for object retrieval. TPAMI (2018)
5. Bai, S., Zhou, Z., Wang, J., Bai, X., Latecki, L.J., Tian, Q.: Ensemble diffusion for retrieval
6. Cao, Y., Long, M., Wang, J., Zhu, H., Wen, Q.: Deep quantization network for efficient image retrieval. In: AAAI (2016)
7. Charikar, M.S.: Similarity estimation techniques from rounding algorithms. In: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing. pp. 380–388 (2002)
8. 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: Proceedings of the ACM international conference on image and video retrieval. p. 48 (2009)
9. Ge, T., He, K., Ke, Q., Sun, J.: Optimized product quantization for approximate nearest neighbor search. In: CVPR. pp. 2946–2953. IEEE (2013)
10. Gong, Y., Lazebnik, S., Gordo, A., Perronnin, F.: Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. IEEE T-PAMI **35**(12), 2916–2929 (2013)
11. Gordo, A., Almazán, J., Revaud, J., Larlus, D.: Deep image retrieval: Learning global representations for image search. In: ECCV. pp. 241–257. Springer (2016)
12. Hong, W., Meng, J., Yuan, J.: Distributed composite quantization. In: AAAI (2018)
13. Hong, W., Meng, J., Yuan, J.: Tensorized projection for high-dimensional binary embedding. In: AAAI (2018)
14. Hong, W., Yuan, J.: Fried binary embedding: From high-dimensional visual features to high-dimensional binary codes. IEEE Transactions on Image Processing **27**(10) (2018)
15. Hong, W., Yuan, J., Bhattacharjee, S.D.: Fried binary embedding for high-dimensional visual features. CVPR **11**, 18 (2017)
16. Jain, H., Zepeda, J., Perez, P., Gribonval, R.: Subic: A supervised, structured binary code for image search. In: ICCV. pp. 833–842 (2017)
17. Jegou, H., Douze, M., Schmid, C.: Product quantization for nearest neighbor search. IEEE T-PAMI **33**(1), 117–128 (2011)
18. Jégou, H., Douze, M., Schmid, C., Pérez, P.: Aggregating local descriptors into a compact image representation. In: CVPR. pp. 3304–3311 (2010)
19. Jiang, Q.Y., Li, W.J.: Asymmetric deep supervised hashing. AAAI (2018)
20. Klein, B., Wolf, L.: In defense of product quantization. arXiv preprint arXiv:1711.08589 (2017)
21. Krizhevsky, A.: Learning multiple layers of features from tiny images (2009)
22. Lai, H., Pan, Y., Liu, Y., Yan, S.: Simultaneous feature learning and hash coding with deep neural networks. arXiv preprint arXiv:1504.03410 (2015)
23. Li, Q., Sun, Z., He, R., Tan, T.: Deep supervised discrete hashing. In: NIPS. pp. 2479–2488 (2017)

24. Li, W.J., Wang, S., Kang, W.C.: Feature learning based deep supervised hashing with pairwise labels. arXiv preprint arXiv:1511.03855 (2015)
25. Liu, H., Wang, R., Shan, S., Chen, X.: Deep supervised hashing for fast image retrieval. In: CVPR. pp. 2064–2072 (2016)
26. Liu, W., Wang, J., Ji, R., Jiang, Y.G., Chang, S.F.: Supervised hashing with kernels. In: CVPR. pp. 2074–2081 (2012)
27. Martinez, J., Clement, J., Hoos, H.H., Little, J.J.: Revisiting additive quantization. In: European Conference on Computer Vision. pp. 137–153. Springer (2016)
28. Ng, J.Y.H., Yang, F., Davis, L.S.: Exploiting local features from deep networks for image retrieval. arXiv preprint arXiv:1504.05133 (2015)
29. Norouzi, M., Fleet, D.J.: Cartesian k-means. In: CVPR. pp. 3017–3024 (2013)
30. Perronnin, F., Liu, Y., Sánchez, J., Poirier, H.: Large-scale image retrieval with compressed fisher vectors. In: CVPR. pp. 3384–3391 (2010)
31. Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.: Object retrieval with large vocabularies and fast spatial matching. In: CVPR. pp. 1–8 (2007)
32. Salakhutdinov, R., Hinton, G.: Semantic hashing. RBM **500**(3), 500 (2007)
33. Shen, F., Shen, C., Liu, W., Shen, H.T.: Supervised discrete hashing. IEEE T-PAMI **35**(12), 2916–2929 (2013)
34. Wang, X., Shi, Y., Kitani, K.M.: Deep supervised hashing with triplet labels. In: ACCV. pp. 70–84. Springer (2016)
35. Wang, X., Zhang, T., Qi, G.J., Tang, J., Wang, J.: Supervised quantization for similarity search. In: CVPR. pp. 2018–2026 (2016)
36. Weiss, Y., Torralba, A., Fergus, R.: Spectral hashing. In: NIPS. pp. 1753–1760 (2009)
37. Xia, R., Pan, Y., Lai, H., Liu, C., Yan, S.: Supervised hashing for image retrieval via image representation learning. In: AAAI. pp. 2156–2162. AAAI Press (2014)
38. Yu, T., Meng, J., Yuan, J.: Is my object in this video? reconstruction-based object search in videos. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence. pp. 4551–4557. AAAI Press (2017)
39. Yu, T., Wang, Z., Yuan, J.: Compressive quantization for fast object instance search in videos. In: ICCV. pp. 833–842 (2017)
40. Yu, T., Wu, Y., Bhattacharjee, S.D., Yuan, J.: Efficient object instance search using fuzzy objects matching. AAAI (2017)
41. Yu, T., Wu, Y., Yuan, J.: Hope: Hierarchical object prototype encoding for efficient object instance search in videos. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2424–2433 (2017)
42. Zhang, R., Lin, L., Zhang, R., Zuo, W., Zhang, L.: Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification. IEEE TIP **24**(12), 4766–4779 (2015)
43. Zhang, T., Du, C., Wang, J.: Composite quantization for approximate nearest neighbor search. In: ICML. pp. 838–846. No. 2 (2014)
44. Zhang, Z., Chen, Y., Saligrama, V.: Efficient training of very deep neural networks for supervised hashing. In: CVPR. pp. 1487–1495 (2016)
45. Zhao, F., Huang, Y., Wang, L., Tan, T.: Deep semantic ranking based hashing for multi-label image retrieval. In: CVPR. pp. 1556–1564 (2015)
46. Zhu, H., Long, M., Wang, J., Cao, Y.: Deep hashing network for efficient similarity retrieval. In: AAAI (2016)