Beyond Product Quantization: Deep Progressive Quantization for Image Retrieval

Lianli Gao[1] , **Xiaosu Zhu**[1] , **Jingkuan Song**[1] , **Zhou Zhao**[2] and **Heng Tao Shen**[1*]

[1]Center for Future Media, University of Electronic Science and Technology of China
[2]Zhejiang University

lianli.gao@uestc.edu.cn, xiaosu.zhu@outlook.com, jingkuan.song@gmail.com, zhaozhou@zju.edu.cn, shenhengtao@hotmail.com

Abstract

Product Quantization (PQ) has long been a mainstream for generating an exponentially large codebook at very low memory/time cost. Despite its success, PQ is still tricky for the decomposition of high-dimensional vector space, and the retraining of model is usually unavoidable when the code length changes. In this work, we propose a deep progressive quantization (DPQ) model, as an alternative to PQ, for large scale image retrieval. DPQ learns the quantization codes sequentially and approximates the original feature space progressively. Therefore, we can train the quantization codes with different code lengths simultaneously. Specifically, we first utilize the label information for guiding the learning of visual features, and then apply several quantization blocks to progressively approach the visual features. Each quantization block is designed to be a layer of a convolutional neural network, and the whole framework can be trained in an end-to-end manner. Experimental results on the benchmark datasets show that our model significantly outperforms the state-of-the-art for image retrieval. Our model is trained once for different code lengths and therefore requires less computation time. Additional ablation study demonstrates the effect of each component of our proposed model. Our code is released at https://github.com/cfm-uestc/DPQ.

1 Introduction

With the rapidly increasing amount of images, similarity search in large-scale image datasets has been an active research topic in many domains, including computer vision and information retrieval [Wang *et al.*, 2018]. However, exact nearest-neighbor (NN) search is often intractable because of the size of dataset and the curse of dimensionality for images. Instead, approximate nearest-neighbor (ANN) search is more practical and can achieve orders of magnitude in speed-up compared to exact NN search [Shakhnarovich *et al.*, 2008; Song *et al.*, 2018b; Song *et al.*, 2018a; Song *et al.*, 2018c; Lew *et al.*, 2006].

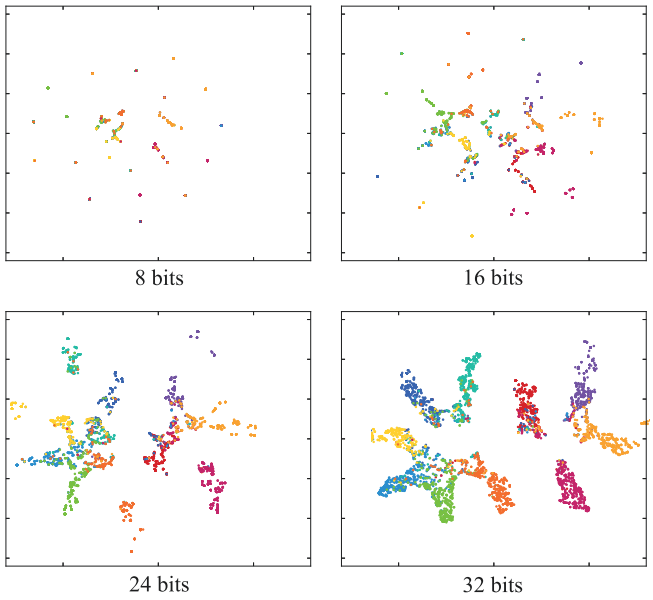


Figure 1: t-SNE visualization of progressively generated quantized features (8, 16, 24 and 32 bits) on CIFAR-10. Different colors indicate different labels.

Recently, hashing methods [Wang *et al.*, 2018; Gionis *et al.*, 1999; Liu *et al.*, 2012; Kulis and Darrell, 2009; Shen *et al.*, 2015; Liu *et al.*, 2016; Zhu *et al.*, 2016; Xia *et al.*, 2014; Lai *et al.*, 2015] are popular for scalable image retrieval due to their compact binary representation and efficient Hamming distance calculation. Such approaches embed data points to compact binary codes by hash functions, while the similarity between vectors are preserved by Hamming distance.

On the other hand, quantization-based methods aim at minimizing the quantization error, and have been shown to achieve superior accuracy [Cao *et al.*, 2017; Zhang *et al.*, 2015] over hashing methods, with sacrifice of efficiency. For input data $\mathbf{X} \in \mathbb{R}^{N\times D}$ which contains N samples of D dimensions, Vector Quantization [Gersho and Gray, 2012] tries to create a codebook $\boldsymbol{C} \in \mathbb{R}^{K\times D}$ which contains K codewords $\mathbf{c}(k) \in \mathbb{R}^{D}, k = 1, 2, \cdots, K$, and assigns each data $\mathbf{x}_i, i = 1, 2, \cdots, N$ to its nearest codeword by a quantizer

*Contact Author

$q(\mathbf{x}_i) = \mathbf{c}(e(\mathbf{x}_i))$. In information theory, the function $e(\cdot)$ is called an encoder, and $\mathbf{c}(\cdot)$ is called a decoder [Gersho and Gray, 2012]. The goal of quantization is to minimize the distortion between raw data and quantized data:

$$min \sum\nolimits_i \|\mathbf{x}_i - q(\mathbf{x}_i)\|_2 \tag{1}$$

After encoding, a D-dimensional data becomes $e(\mathbf{x}_i) \in [1, 2, ..., K]$, which can be represented as a compact binary code $\mathbf{b}_i$ of length $\log_2 K$. Product Quantization [Jegou *et al.*, 2011] and Optimized Product Quantization [Ge *et al.*, 2013] first divide the feature space $\mathbb{R}^D$ to M subspaces (OPQ also performs a rotation on the data) and perform previous minimization on each subspace. Obviously, the codebooks increase from 1 to M. Composite Quantization [Zhang *et al.*, 2015] also learns M codebooks, but its codewords have the same dimension as the original features. Similar to CQ, Stacked Quantization [Martinez *et al.*, 2014] also uses the sum of multiple codewords to approximate the raw data. But Stacked Quantization uses residual of the quantized results and proposes a hierarchical structure so that the minimization can be performed on each codebook. By integrating deep learning to quantization methods, Cao *et al*. proposed Deep Quantization Network [Cao *et al.*, 2016]. It is the first deep learning structure that learns feature by pairwise cosine distance. Then, Cao *et al*. proposed Deep Visual-Semantic Quantization [Cao *et al.*, 2017] which projects the feature space to semantic space. Inspired by NetVLAD [Arandjelovic *et al.*, 2016], Benjamin *et al*. proposed Deep Product Quantization [Klein and Wolf, 2017] that enabled differentiable quantization.

To generate an exponentially large codebook at very low memory/time cost, a product quantizer is still the first option. Despite its success, PQ and its variants have several issues. First, to generate binary codes with different code lengths, a retraining is usually unavoidable. Second, it is still tricky for the decomposition of high-dimensional vector space. Different decomposition strategies may result in huge performance differences. To address these issues, we propose a deep progressive quantization (DPQ) model, as an alternative to PQ, for large scale image retrieval. The contributions of DPQ can be summarized as: 1) DPQ is a general framework which can learn codes with different lengths simultaneously, by approximating the original feature space progressively. Different components of DPQ are replaceable and we instantiate it with a few specific designs of label utilization, input and quantization blocks; 2) Each component in the framework is designed to be differentiable, and thus the whole framework can be trained in an end-to-end manner; and 3) Extensive experiments on the benchmark dataset show that DPQ significantly outperforms the state-of-the-art for image retrieval. Additional ablation study demonstrates the effect of each component of our model.

2 Proposed Method

Given N images $\mathbf{I}$ and a query $\mathbf{q}$, the NN search problem aims to find the item NN($\mathbf{q}$) from $\mathbf{I}$ such that its distance to the query is minimum. We study the ANN search problem and we propose a coding approach called Deep Progressive Quantization (DPQ) for fast image retrieval.

The idea is to approximate a vector $\mathbf{x}$ by the composition of several (L) elements $\left\{\mathbf{c}^1(k_1), \mathbf{c}^2(k_2), ..., \mathbf{c}^L(k_L)\right\}$, each of which is selected from a dictionary $\mathbf{C}^l$ with K elements, e.g., $\mathbf{c}^1(k_1)$ is the k_1-th element in dictionary $\mathbf{C}^1$, and to represent a vector by a short code composed of the indices of selected elements, resulting in a binary codes of length $L \cdot \log_2 K$ with each element coded by $\log_2 K$ bits.

We illustrate our proposed DPQ architecture by the scheme in Fig. 2. It consists of two major components: (1) a supervised feature extraction component in a multi-task fashion and (2) a Deep Progressive Quantization component to convert the features to binary codes. In the remainder of this section, we first describe each of them and then illustrate the optimization and search process.

2.1 Feature Extraction using Multi-task Learning

For feature extraction, we use AlexNet [Krizhevsky *et al.*, 2012], and we use the output of $fc7$ layer to extract 4096-d features. To fully utilize the supervision information, we introduce two losses based on labels. The first one is a traditional classification loss, which is defined as:

$$L_C(\mathbf{x}) = -\sum_i y'_i \log(y_i) \tag{2}$$

$$y'_i = softmax(\text{MLP}(\mathbf{x})) \tag{3}$$

where $\mathbf{x}$ is the input feature, y_i is the label and y'_i is the predicted label. When our images have multi-labels, we modify this loss by multi-label sigmoid cross-entropy:

$$L_C(\mathbf{x}) = -(y'_i) \times y_i + \log\left(1 + e^{y'_i}\right) \tag{4}$$

$$y'_i = \text{MLP}(\mathbf{x}) \tag{5}$$

The second one is an adaptive margin loss. We first embed each label into a 300-d semantic vector $\mathbf{z}_n$ using word-embedding [Cao *et al.*, 2017]. By applying a fully connected layer on the 4096-d features, we can predict the semantic embedding $\mathbf{v}_i$ of each data. Following DVSQ [Cao *et al.*, 2017], to embed the projects with the semantic label, an adaptive margin loss is defined as:

$$L_S(x) = \sum_{i \in \mathcal{Y}_n} \sum_{j \notin \mathcal{Y}_n} (0, \delta_{ij} - \frac{\boldsymbol{v}_i^\intercal \boldsymbol{z}_n}{\|\boldsymbol{v}_i\| \|\boldsymbol{z}_n\|} + \frac{\boldsymbol{v}_j^\intercal \boldsymbol{z}_n}{\|\boldsymbol{v}_j\| \|\boldsymbol{z}_n\|}) \tag{6}$$

$$\delta_{ij} = 1 - \frac{\boldsymbol{v}_i^\intercal \boldsymbol{v}_j}{\|\boldsymbol{v}_i\| \|\boldsymbol{v}_j\|} \tag{7}$$

These two branches can be described as two different tasks.

2.2 Deep Progressive Quantization

The structure of DPQ is shown in Fig. 2. We denote $\mathbf{x} \in \mathbb{R}^D$ as the features of an input image. It goes through L quantization layers (Q-Block), and each Q-Block outputs m-bit quantization codes $\mathbf{b}^l$ and quantized value $\mathbf{q}^l$. The quantization process can be formulated as:

$$\mathbf{q}^1 = Q^1\left(\mathbf{x}^1\right), \mathbf{q}^2 = Q^2\left(\mathbf{x}^2\right), ..., \mathbf{q}^L = Q^L\left(\mathbf{x}^L\right) \tag{8}$$

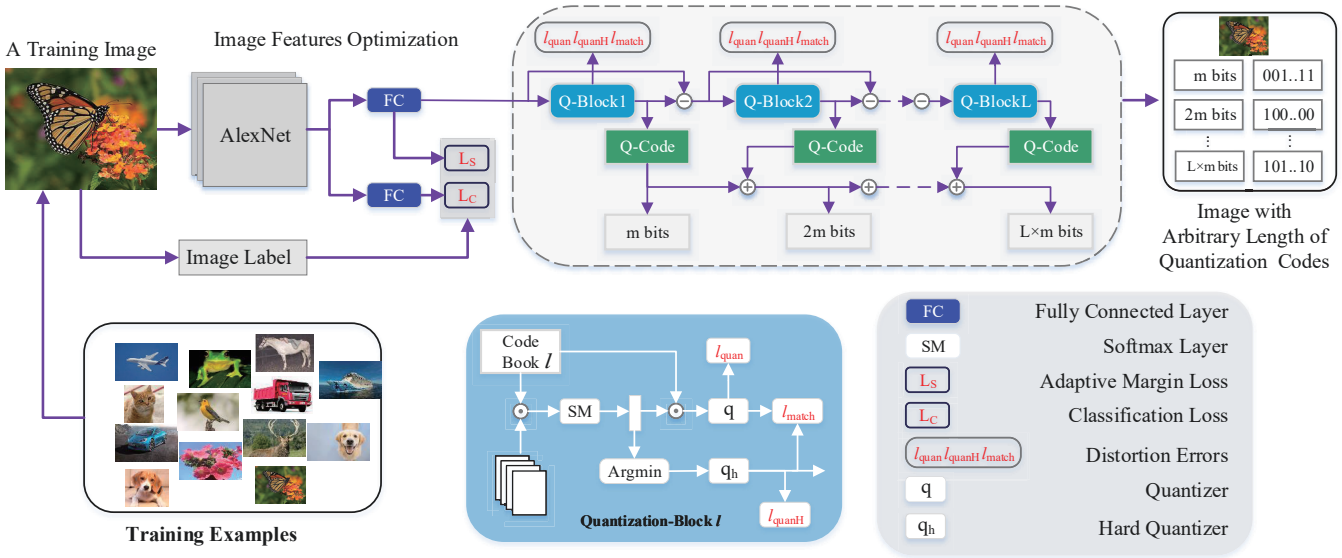


Figure 2: Illustration of our Deep Progressive Quantization for fast image retrieval.

where $\mathbf{x}^l$ is the input of the l-th quantization layer, and it is quantized to $\mathbf{q}^l$ by the l-th quantizer Q^l.

The target of DPQ is to progressively approximate $\mathbf{x}$ by $\sum_l \mathbf{q}^i$, and objective function can be formulated as:

$$\begin{cases} \ell^1_{\text{quan}} = \mathbf{x} - \mathbf{q}^1 \\ \ell^2_{\text{quan}} = \mathbf{x} - \sum_{l=1}^{2} \mathbf{q}^l \\ \quad\vdots \\ \ell^L_{\text{quan}} = \mathbf{x} - \sum_{l=1}^{L} \mathbf{q}^i \end{cases} \tag{9}$$

The final objective function can be formulated as a weighted sum of quantization losses:

$$\ell_{\text{quan}} = w^1\ell^1_{\text{quan}} + w^2\ell^2_{\text{quan}} + ... + w^L\ell^L_{\text{quan}} \tag{10}$$

An important property of this quantization algorithm is that it can generate hash codes with different code lengths simultaneously. In the l-th Q-Block, the sum of the $1 \sim l$ quantized value can approximate the original feature $\mathbf{x}$, as shown in Eq. 9. Note that this is a general form for Deep Progressive Quantization. To instantiate DPQ, we need to specify the design of each quantizer Q^l and its input $\mathbf{x}^l$.

A straightforward candidate for the Q-Block is K-means algorithm. Given N D-dimensional points $\mathbf{X}$, the K-means algorithm partitions the database into K clusters, each of which associates one codeword $\mathbf{c}_k \in \mathbb{R}^D$. Let $\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_K] \in \mathbb{R}^{D\times K}$ be the corresponding codebook. K-means first randomly initializes a $\mathbf{C}$, and then each data point $\mathbf{x}_i$ is quantized to:

$$Q_{\text{Kmeans}}(\mathbf{x}_i) = \underset{\mathbf{c}(e(\mathbf{x}_i))}{\arg\min} \|\mathbf{x} - \mathbf{c}(e(\mathbf{x}_i))\|_2, k = 1,2,\cdots,K \tag{11}$$

Then the codebook is learned by minimizing the within cluster distortion, i.e.,

$$\min_{\mathbf{C}} \sum_i \|\mathbf{x}_i - Q_{\text{Kmeans}}(\mathbf{x}_i)\|_2 \tag{12}$$

This optimization can be done by Lloyd algorithm. However, if we directly use K-means as our Q-Block, it is not an end-to-end model, because $Q_{\text{Kmeans}}(\mathbf{x}_i)$ is *non-differentiable* to $\mathbf{c}(i)$.

To integrate the traditional quantization method into the deep architecture, we have to design a differentiable quantization function. Inspired by NetVLAD [Arandjelovic *et al.*, 2016], we observe that Eq. 11 is equivalent to:

$$Q_{\text{Kmeans}}(\mathbf{x}_i) = \lim_{\gamma\to+\infty} \sum_k \frac{e^{-\gamma\|\mathbf{x}_i - \mathbf{c}(k)\|_2}}{\sum_k e^{-\gamma\|\mathbf{x}_i - \mathbf{c}(k)\|_2}} \mathbf{c}(k) \tag{13}$$

When $\gamma \to +\infty$, $\mathbf{x}_i$ is quantized to its closest cluster center $\mathbf{c}(e(\mathbf{x}_i))$. In implementation, choosing a large γ can approximate $Q_{\text{Kmeans}}(\mathbf{x})$ well. Therefore, we design our quantization function as:

$$Q_{\text{DPQ}}(\mathbf{x}) = \sum_k \frac{e^{-\gamma d(\mathbf{x},\mathbf{c}(k))}}{\sum_k e^{-\gamma d(\mathbf{x},\mathbf{c}(k))}} \mathbf{c}(k) \tag{14}$$

where $d(\mathbf{x}, \mathbf{c}(k))$ indicates the distance between a vector $\mathbf{x}$ and a cluster center $\mathbf{c}(k)$. Importantly, the proposed quantization function is continuously *differentiable*, which can be readily embedded to any neural networks. $d(\mathbf{x}, \mathbf{c}(k))$ can be any distance function. In practice, we find cosine similarity is better than Euclidean distance, and we define it as:

$$d(\mathbf{x}, \mathbf{c}(k)) = -\frac{\langle \mathbf{x}, \mathbf{c}(k)\rangle}{\|\mathbf{x}\|\|\mathbf{c}(k)\|_2}, i = 1, 2, \cdots, K \tag{15}$$

where $\langle\cdot,\cdot\rangle$ denotes the inner product between two vectors.

We then focus on the input of Q-Block. Following the composition in [Martinez *et al.*, 2014] and based on Eq. 8, $\mathbf{q}^1$ can be interpreted as a quantizer which approximates the input. But the other quantizers, i.e., q^l, can be interpreted as quantizers to approximate the residual of the previous l-1 quantizers. It is not required that $\mathbf{q}^l \approx \mathbf{x}^l$. The solution is to let:

$$\begin{aligned} \mathbf{x}^1 &= \mathbf{x} \\ \mathbf{x}^l &= \mathbf{x}^{l-1} - \mathbf{q}^{l-1}, l > 1 \end{aligned} \tag{16}$$

q^l can be interpreted as a quantizer to approximate the input.

After we determine the quantization functions (Eq. 14) and their inputs (Eq. 16), we can calculate the quantization loss

based on Eq. 8, Eq. 9 and Eq. 10. However, the quantization functions (Eq. 14) is not a hard assignment, i.e., $Q_{\text{DPQ}}(\mathbf{x})$ is a linear combination of all codewords in codebook. While during the retrieval, each vector is quantized to its closest codeword, named hard assignment, defined as:

$$Q^H_{\text{DPQ}}(\mathbf{x}) = \underset{\mathbf{c}(e(\mathbf{x}))}{\arg\min} \|\mathbf{x} - \mathbf{c}(e(\mathbf{x}))\|_2, k = 1, 2, \cdots, K \quad (17)$$

Therefore, we further define the quantization losses based on the hard assignment as:

$$\begin{cases} \ell^1_{\text{quanH}} = \mathbf{x} - \mathbf{q}^1_H \\ \ell^2_{\text{quanH}} = \mathbf{x} - \sum_{l=1}^{2} \mathbf{q}^l_H \\ \quad\vdots \\ \ell^L_{\text{quanH}} = \mathbf{x} - \sum_{l=1}^{L} \mathbf{q}^L_H \end{cases} \quad (18)$$

$$\ell_{\text{quanH}} = w^1 \ell^1_{\text{quanH}} + w^2 \ell^2_{\text{quanH}} + ... + w^L \ell^L_{\text{quanH}} \quad (19)$$

where $\mathbf{q}^l_H$ is the hard quantization value of $\mathbf{x}$ in the l-th layer. We further constrain that the learned $\mathbf{q}^l$ should be similar to the hard assignment $\mathbf{q}^l_H$, formulated as:

$$\ell_{\text{match}} = w^1 \|\mathbf{q}^1, \mathbf{q}^1_H\| + w^2 \|\mathbf{q}^2, \mathbf{q}^2_H\| + ... + w^L \|\mathbf{q}^L, \mathbf{q}^L_H\| \quad (20)$$

Therefore, the overall distortion loss function is:

$$E(\boldsymbol{x}) = \ell_{\text{quan}} + \mu \ell_{\text{quanH}} + \nu \ell_{\text{match}} \quad (21)$$

2.3 Optimization

We train our network by optimizing loss functions (L_S, L_C, E). The pipeline of the whole training can be described as follows: Firstly, we collect a batch of training images $\boldsymbol{I}$ as inputs of CNN and extract their features $\boldsymbol{X}$. We then use Q-Block to obtain its quantized features $\boldsymbol{q}_i$, $\boldsymbol{q}_{H,i}$ and associated binary code $\boldsymbol{b}_i$:

$$\boldsymbol{X} = \Theta(\boldsymbol{I}; \theta),\ \boldsymbol{q}^l = Q^l_{\text{DPQ}}(\boldsymbol{x}; \pi), \boldsymbol{q}^l_H = Q^l_H(\boldsymbol{x}; \pi) \quad (22)$$

$$\boldsymbol{b}^l \leftarrow \underset{e(\mathbf{x})}{\arg\min}\, d(\mathbf{x}, \mathbf{c}(e(\mathbf{x}))), e(\mathbf{x}) = 1, 2, \cdots, K \quad (23)$$

where θ and π are the parameters of CNN and Q-Block.

Now, we can use the outputs to calculate total loss function:

$$L(\boldsymbol{x}; \theta, \pi) = L_S(\boldsymbol{x}) + \lambda L_C(\boldsymbol{x}) + \tau E(\boldsymbol{x}) \quad (24)$$

Note that the loss function has parameters θ and π to be learned, we use mini-batch gradient descent to update parameters and minimize the loss:

$$\theta \leftarrow \theta - \eta \nabla_\theta (L_S, L_C, E),\ \pi \leftarrow \pi - \eta \nabla_\pi (E) \quad (25)$$

where η is the learning rate. We train our network until the maximum iteration is reached.

2.4 Retrieval Procedure

After the model is learned, we will obtain a codebook π. Next, we need to encode the database and perform the search. Given a data point $\mathbf{x}_i$ in the database, the l-th Q-Block quantizes the input $\mathbf{x}^l_i$ to:

$$e(\mathbf{x}^l_i) = \underset{k^l}{\arg\min} \|\mathbf{x}^l_i - \mathbf{c}^l(k^l)\|^2_2, k^l = 1, 2, \cdots, K \quad (26)$$

$$Q^H_{\text{DPQ}}(\mathbf{x}^l_i) = \mathbf{c}^l(e(\mathbf{x}^l_i)) \quad (27)$$

And therefore, $\mathbf{x}_i$ is quantized by $\overline{\mathbf{x}}_i$ and represented by $\mathbf{b}_i$:

$$\overline{\mathbf{x}}_i = \sum\nolimits_l Q^H_{\text{DPQ}}(\mathbf{x}^l_i) = \sum\nolimits_l \mathbf{c}^l(e(\mathbf{x}^l_i)) \quad (28)$$

$$\mathbf{b}_i = [\text{binary}(e(\mathbf{x}^1_i)), \text{binary}(e(\mathbf{x}^2_i)), .., \text{binary}(e(\mathbf{x}^L_1))] \quad (29)$$

where binary$(.)$ operation is to convert an index in the range of $[1, K]$ to a binary code with the code length of M=$\log_2 K$. Therefore, we can simultaneously obtain L binary codes with different lengths.

The Asymmetry Quantization Distance (AQD) of a query $\mathbf{q}$ to database point $\mathbf{x}_i$ is approximated by $\|\mathbf{q} - \overline{\mathbf{x}}\|^2_2 = \left\|\mathbf{q} - \sum_l \mathbf{c}^l(e(\mathbf{x}^l_i))\right\|^2_2$, which can be reformulated as:

$$\|q - \sum_l \mathbf{c}^l(e(\mathbf{x}^l_i))\|^2_2 = \sum_l \|\mathbf{q} - \mathbf{c}^l(e(\mathbf{x}^l_i))\|^2_2 \quad (30)$$

$$-(L-1)\|\mathbf{q}\|^2_2 + \sum_{l1=1}^{L} \sum_{l2=1, l2\neq l1}^{L} \left(\mathbf{c}^{l1}(e(\mathbf{x}^{l1}_i))\right)^T \mathbf{c}^{l2}(e(\mathbf{x}^{l2}_i))$$

We can see that the first and third term can be precomputed and stored, while the second term is constant. Therefore, the AQD can be efficiently calculated.

3 Experiments

We evaluate our DPQ on the task of image retrieval. Specifically, we first evaluate the effectiveness of DPQ against the state-of-the-art methods on three datasets, and then conduct ablation study to explore the effect of important components.

3.1 Setup

We conduct the experiments on three public benchmark datasets: **CIFAR-10**, **NUS-WIDE** and **ImageNet**.

CIFAR-10 [Krizhevsky and Hinton, 2009] is a public dataset labeled in 10 classes. It consists of 50,000 images for training and 10,000 images for validation. We follow [Cao *et al.*, 2016; Cao *et al.*, 2017] to combine all images together. Randomly select 500 images per class as the training set, and 100 images per class as the query set. The remaining images are used as database.

NUS-WIDE [Chua *et al.*, 2009] consists of 81 concepts, and each image is annotated with one or more concepts. We follow [Cao *et al.*, 2016; Cao *et al.*, 2017] to use the subset of the 21 most frequent concepts (195,834 images). We randomly sample 5,000 images as the query set, and use the remaining images as database. Furthermore, we randomly select 10,000 images from the database as the training set.

ImageNet [Deng *et al.*, 2009] contains 1.2M images labeled with 1,000 classes. We follow [Cao *et al.*, 2017] to randomly choose 100 classes. We use all images of these classes in the training set and validation set as the database and queries respectively. We randomly select 100 images for each class in the database for training.

We compare our method with state-of-the-art supervised hashing and quantization methods, including 5 shallow-based methods (**KSH** [Liu *et al.*, 2012], **SDH** [Shen *et al.*, 2015], **BRE** [Kulis and Darrell, 2009], **ITQ-CCA** [Gong *et al.*, 2013] and **SQ** [Martinez *et al.*, 2014]) and 6 deep-based methods (**DSH** [Liu *et al.*, 2016], **DHN** [Zhu *et al.*,

Method	CIFAR-10				NUS-WIDE				ImageNet			
	8 bits	16 bits	24 bits	32 bits	8 bits	16 bits	24 bits	32 bits	8 bits	16 bits	24 bits	32 bits
ITQ-CCA [Gong *et al.*, 2013]	0.315	0.354	0.371	0.414	0.526	0.575	0.572	0.594	0.189	0.270	0.339	0.436
BRE [Kulis and Darrell, 2009]	0.306	0.370	0.428	0.438	0.550	0.607	0.605	0.608	0.251	0.363	0.404	0.453
KSH [Liu *et al.*, 2012]	0.489	0.524	0.534	0.558	0.618	0.651	0.672	0.682	0.228	0.398	0.499	0.547
SDH [Shen *et al.*, 2015]	0.356	0.461	0.496	0.520	0.645	0.688	0.704	0.711	0.385	0.516	0.570	0.605
SQ [Martinez *et al.*, 2014]	0.567	0.583	0.602	0.615	0.653	0.691	0.698	0.716	0.465	0.536	0.592	0.611
CNNH [Xia *et al.*, 2014]	0.461	0.476	0.465	0.472	0.586	0.609	0.628	0.635	0.317	0.402	0.453	0.476
DNNH [Lai *et al.*, 2015]	0.525	0.559	0.566	0.558	0.638	0.652	0.667	0.687	0.347	0.416	0.497	0.525
DHN [Zhu *et al.*, 2016]	0.512	0.568	0.594	0.603	0.668	0.702	0.713	0.716	0.358	0.426	0.531	0.556
DSH [Liu *et al.*, 2016]	0.592	0.625	0.651	0.659	0.653	0.688	0.695	0.699	0.332	0.398	0.487	0.537
DQN [Cao *et al.*, 2016]	0.527	0.551	0.558	0.564	0.721	0.735	0.747	0.752	0.488	0.552	0.598	0.625
DVSQ [Cao *et al.*, 2017]	0.715	0.727	0.730	0.733	0.780	0.790	0.792	0.797	0.500	0.502	0.505	0.518
DPQ	**0.814**	**0.833**	**0.834**	**0.831**	**0.786**	**0.821**	**0.832**	**0.834**	**0.521**	**0.602**	**0.613**	**0.623**

Table 1: Quantitative comparison with state-of-the-art methods on on three datasets. The scores reported are mean Average Precision values.

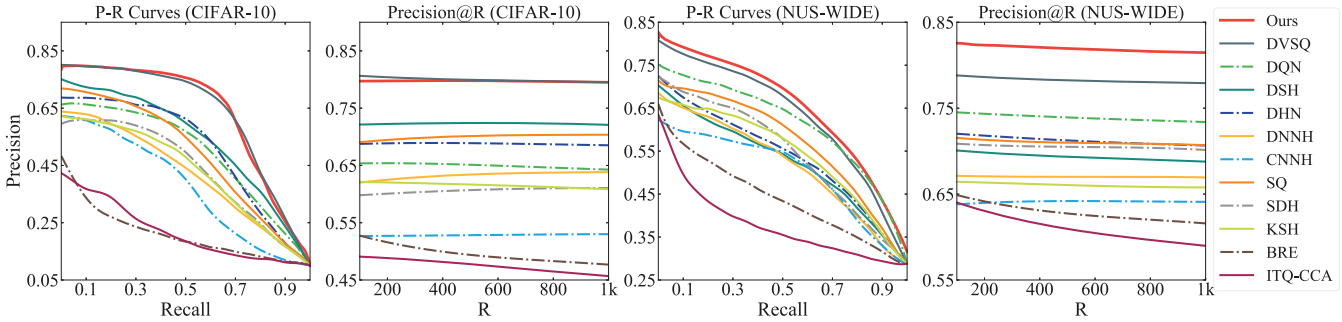


Figure 3: Quantitative comparison with state-of-the-art methods on two datasets: CIFAR-10 and NUS-WIDE. For each data set, we demonstrate Precision-Recall (P-R) curves and Precision@R curves. All results are based on 32-bit.

2016], **DNNH** [Lai *et al.*, 2015], **CNNH** [Xia *et al.*, 2014], **DQN** [Cao *et al.*, 2016] and **DVSQ** [Cao *et al.*, 2017]). For shallow-based methods, we use $fc7$ of pre-trained AlexNet as features. We evaluate all methods with 8, 16, 24 and 32 bits.

To evaluate retrieval effectiveness on these datasets, we follow [Cao *et al.*, 2016; Cao *et al.*, 2017; Lai *et al.*, 2015] to use three evaluation metrics: mean Average Precision (mAP), Precision-Recall curves, and Precision@R (number of returned samples) curves. We follow the settings in [Cao *et al.*, 2016; Cao *et al.*, 2017] to measure mAP@54000 on CIFAR-10, mAP@5000 on NUS-WIDE and mAP@5000 on ImageNet. Following [Zhu *et al.*, 2016; Liu *et al.*, 2016; Cao *et al.*, 2016; Cao *et al.*, 2017], the evaluation is based on Asymmetric Quantization Distance (AQD). We use 300-d features as the input of the Q-Block. In order to compare with other methods using the same bit-length, we construct the codebook with $L = 4$, K=256=2^8, so that each codebook can provide a piece of 8-bit binary code. We set epoch to 64 and batch to 16. We use the Adam optimizer with default value. We tune the learning rate η from 10^{-4} to 10^{-1}. As for λ, τ, μ, ν in loss function Eq. 24, we empirically set them as $\lambda = 0.1, \tau = 1, \mu = 1, \nu = 0.1$. Our implementation is based on Tensorflow.

3.2 Comparison With the State-of-the-Art Methods

The mAP of compared methods are based on DVSQ [Cao *et al.*, 2017], and the results are shown in Tab. 1. It can be observed that: 1) Our method (DPQ) significantly outperforms the other deep and non-deep hashing methods in all datasets. In CIFAR-10, the improvement of DPQ over the other methods is more significant, compared with that in NUS-WIDE and ImageNet datasets. Specifically, it outperforms the best counterpart (DVSQ) by 9.9%, 10.6%, 10.4% and 9.8% for 8, 16, 24 and 32-bits hash codes. DPQ improves the state-of-the-art by 0.6%, 3.1%, 4.0% and 3.7% in NUS-WIDE dataset, and 2.1%, 6.6%, 2.1%, 1.2% in ImageNet dataset. 2) With the increase of code length, the performance of most indexing methods is improved accordingly. For our DPQ, the mAP increased by 2.0%, 4.8% and 10.2% for CIFAR-10, NUS-WIDE and ImageNet dataset respectively. The improvement for CIFAR-10 dataset is relatively small. One possible reason is that CIFAR-10 contains simple images, and short codes are good enough to conduct accu-

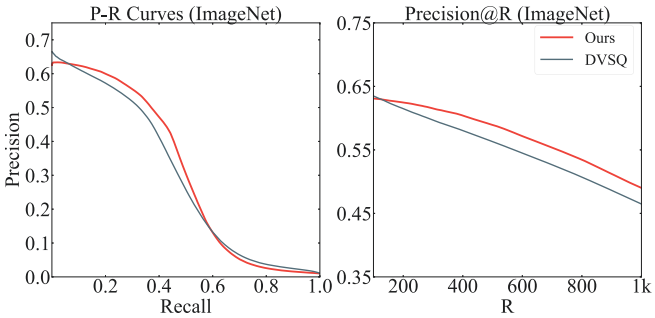
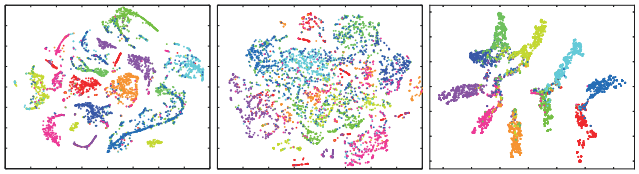


Figure 4: Quantitative comparison with DVSQ method on ImageNet. We demonstrate Precision-Recall (P-R) curves and Precision@R curves. All results are based on 32-bit.

Method	CIFAR-10				NUS-WIDE				ImageNet			
	8 bits	16 bits	24 bits	32 bits	8 bits	16 bits	24 bits	32 bits	8 bits	16 bits	24 bits	32 bits
E	0.172	0.188	0.184	0.205	0.472	0.620	0.684	0.713	0.248	0.265	0.277	0.283
$E + L_S$	0.741	0.750	0.755	0.764	0.780	0.815	0.823	0.823	0.514	0.534	0.540	0.543
$E + L_C$	0.614	0.644	0.654	0.671	0.475	0.580	0.616	0.661	0.328	0.393	0.408	0.418
Two-Step	0.732	0.755	0.758	0.760	0.505	0.626	0.656	0.700	0.345	0.399	0.419	0.412
No-Soft	0.556	0.575	0.598	0.621	0.542	0.684	0.701	0.692	0.302	0.332	0.320	0.328
DPQ	**0.814**	**0.833**	**0.834**	**0.831**	**0.786**	**0.821**	**0.832**	**0.834**	**0.521**	**0.602**	**0.613**	**0.623**

Table 2: Ablation Studies: mean Average Precision scores of 6 variants of DPQ on three datasets by setting the quantization code length as 8 bits, 16 bits, 24 bits, 32 bits. Variants are described in sec. 3.3.



(a) DQN (b) DVSQ (c) Ours

Figure 5: t-SNE visualization of 32 bits quantized features between DQN, DVSQ and DPQ. Images are randomly sampled on CIFAR-10's database, and samples with different labels are marked with different colors.

rate retrieval. 3) Deep-based quantization methods perform better than shallow-based methods in general. This indicates that jointly learning procedure can usually obtain better features. On the other hand, the performances of two deep hashing methods (CNNH [Xia *et al.*, 2014] and DNNH [Lai *et al.*, 2015]) are unsatisfactory. A possible reason is that the deep hashing methods use only a few fully connected layers to extract the features, which is not very powerful.

The precision-recall curves and precision@R curves for CIFAR-10 and NUS-WIDE datasets are shown in Fig. 3, and the results for ImageNet dataset are shown in Fig. 4. In general, curves for different methods are consistent with their performances of mAP.

3.3 Ablation Study

In this subsection, we conduct a series of ablation studies on all three datasets to see the effect of each component. Specifically, we compare our method with the following changes: 1) E: which removes L_S and L_C from our model. The quantization is directly conducted on $fc7$ from pre-trained AlexNet; 2) $E + L_S$: which remove the L_S loss; 3) $E + L_C$, which removes the L_S loss; 4) Two-Step: which learns features and conducts quantization separately; 5) No-Soft: which removes ℓ_{quanH} and ℓ_{match} from Eq. 21, and optimize hard assignments. The results are shown in Tab. 2.

From Tab. 2, it can be observed that the best performance is achieved by using all the components of our model DPQ. Compared with E which is an unsupervised model, our DPQ improves the mAP by 64.2%, 64.5%, 65.0%, 62.6% for CIFAR-10 dataset, 31.4%, 20.1%, 14.8%, 12.1% for NUS-WIDE dataset and 27.3%, 33.7%, 33.6%, 34.0% for ImageNet dataset. This indicates that the supervised information is very important for learning good representation. $E + L_S$ is a strong competitor, and its performance is very close to DPQ in NUS-WIDE dataset. However, for CIFAR-10 and ImageNet dataset, it is outperformed by DPQ with a large margin. This indicates that L_C is very helpful to guide the learning of features. On the other hand, $E + L_C$, which also utilizes the supervision information by L_C, is not as good as $E + L_C$. This implies that L_S is superior to L_C for retrieval task. Unsurprisingly, DPQ outperforms Two-Step by large margin, especially in NUS-WIDE and ImageNet dataset. This indicates that in Two-Step method, suboptimal hash codes may be produced, since the quantization error is not statistically minimized and the feature representation is not optimally compatible with the quantization process. A joint learning of visual features and hash code can achieve better performance. Compared with No-Soft which only defines the loss function on the hard assignment, DPQ has significant performance gain. Soft-assignment plays the role of intermediate layer, and applying a loss function to soft-assignment is beneficial for hash codes learning.

3.4 Qualitatively Results

We randomly select 5,000 images from CIFAR-10 database and perform t-SNE visualization. Fig. 5 shows the results of DQN, DVSQ and DPQ. As a quantization method, the target is to cluster the data points with the same label, and separate the data points with different labels. Obviously, our DPQ performs the best compared with DQN and DVSQ. For both DQN and DVSQ, the data points with the same label may form several clusters which are far away. This may result in a low recall. On the other hand, the overlapping between different clusters in DQN and DVSQ is more serious than that of DPQ. This may cause a low retrieval precision of DQN and DVSQ, which is also reflected in Tab. 1.

4 Conclusion

In this work, we propose a deep progressive quantization (DPQ) model, as an alternative to PQ, for large scale image retrieval. DPQ learns the quantization code sequentially, and approximates the original feature space progressively. Therefore, we can train the quantization codes with different code lengths simultaneously. Experimental results on the benchmark dataset show that our model significantly outperforms the state-of-the-art for image retrieval.

Acknowledgements

This work is supported by the Fundamental Research Funds for the Central Universities (Grant No. ZYGX2014J063, No. ZYGX2016J085), the National Natural Science Foundation of China (Grant No. 61772116, No. 61872064, No. 61632007, No. 61602049).

References

[Arandjelovic *et al.*, 2016] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *CVPR*, pages 5297–5307, 2016.

[Cao *et al.*, 2016] Yue Cao, Mingsheng Long, Jianmin Wang, Han Zhu, and Qingfu Wen. Deep quantization network for efficient image retrieval. In *AAAI*, pages 3457–3463, 2016.

[Cao *et al.*, 2017] Yue Cao, Mingsheng Long, Jianmin Wang, and Shichen Liu. Deep visual-semantic quantization for efficient image retrieval. In *CVPR*, volume 2, 2017.

[Chua *et al.*, 2009] 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, 2009.

[Deng *et al.*, 2009] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009.

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

[Gersho and Gray, 2012] Allen Gersho and Robert M Gray. *Vector quantization and signal compression*, volume 159. Springer Science & Business Media, 2012.

[Gionis *et al.*, 1999] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. Similarity search in high dimensions via hashing. In *VLDB*, volume 99, pages 518–529, 1999.

[Gong *et al.*, 2013] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2916–2929, 2013.

[Jegou *et al.*, 2011] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2011.

[Klein and Wolf, 2017] Benjamin Klein and Lior Wolf. In defense of product quantization. *arXiv preprint arXiv:1711.08589*, 2017.

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

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.

[Kulis and Darrell, 2009] Brian Kulis and Trevor Darrell. Learning to hash with binary reconstructive embeddings. In *NIPS*, pages 1042–1050, 2009.

[Lai *et al.*, 2015] Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan. Simultaneous feature learning and hash coding with deep neural networks. In *CVPR*, pages 3270–3278, 2015.

[Lew *et al.*, 2006] Michael S Lew, Nicu Sebe, Chabane Djeraba, and Ramesh Jain. Content-based multimedia information retrieval: State of the art and challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 2(1):1–19, 2006.

[Liu *et al.*, 2012] Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. Supervised hashing with kernels. In *CVPR*, pages 2074–2081, 2012.

[Liu *et al.*, 2016] Haomiao Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen. Deep supervised hashing for fast image retrieval. In *CVPR*, pages 2064–2072, 2016.

[Martinez *et al.*, 2014] Julieta Martinez, Holger H Hoos, and James J Little. Stacked quantizers for compositional vector compression. *arXiv preprint arXiv:1411.2173*, 2014.

[Shakhnarovich *et al.*, 2008] Gregory Shakhnarovich, Trevor Darrell, and Piotr Indyk. Nearest-neighbor methods in learning and vision. *IEEE Transactions on Neural Networks*, 19(2):377, 2008.

[Shen *et al.*, 2015] Fumin Shen, Chunhua Shen, Wei Liu, and Heng Tao Shen. Supervised discrete hashing. In *CVPR*, pages 37–45, 2015.

[Song *et al.*, 2018a] Jingkuan Song, Lianli Gao, Li Liu, Xiaofeng Zhu, and Nicu Sebe. Quantization-based hashing: a general framework for scalable image and video retrieval. *Pattern Recognition*, 75:175 – 187, 2018.

[Song *et al.*, 2018b] Jingkuan Song, Tao He, Lianli Gao, Xing Xu, Alan Hanjalic, and Heng Tao Shen. Binary generative adversarial networks for image retrieval. In *AAAI*, pages 394–401, 2018.

[Song *et al.*, 2018c] Jingkuan Song, Hanwang Zhang, Xiangpeng Li, Lianli Gao, Meng Wang, and Richang Hong. Self-supervised video hashing with hierarchical binary auto-encoder. *IEEE Transactions on Image Processing*, 27(7):3210–3221, 2018.

[Wang *et al.*, 2018] Jingdong Wang, Ting Zhang, Nicu Sebe, Heng Tao Shen, et al. A survey on learning to hash. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):769–790, 2018.

[Xia *et al.*, 2014] Rongkai Xia, Yan Pan, Hanjiang Lai, Cong Liu, and Shuicheng Yan. Supervised hashing for image retrieval via image representation learning. In *AAAI*, volume 1, page 2, 2014.

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

[Zhu *et al.*, 2016] Han Zhu, Mingsheng Long, Jianmin Wang, and Yue Cao. Deep hashing network for efficient similarity retrieval. In *AAAI*, pages 2415–2421, 2016.