DEEP PRODUCT QUANTIZATION MODULE FOR EFFICIENT IMAGE RETRIEVAL

Meihan Liu[⋆†‡] *Yongxing Dai*[⋆†] *Yan Bai*[⋆†] *Ling-Yu Duan* [⋆†]

[⋆] Institute of Digital Media, Peking University, China
[†] Peng Cheng Laboratory, Shenzhen, China
[‡] The SECE of Shenzhen Graduate School, Peking University, Shenzhen, China

ABSTRACT

Product Quantization (PQ) is one of the most popular Approximate Nearest Neighbor (ANN) methods for large-scale image retrieval, bringing better performance than hashing based methods. In recent years, several works extend the hard quantization to soft quantization with specially designed deep neural architectures. We propose a simple but effective deep Product Quantization Module (PQM) to jointly learn discriminative codebook and precise hard assignment in an end-to-end manner. In this work, we use the straight-through estimator to make it feasible to directly optimize the discrete binary representations in deep neural networks with stochastic gradient descent. Different from previous deep vector quantization methods, PQM is a plug-and-play module which can be adaptive to various base networks in the scenarios of image search or compression. Besides, we propose a reconstruction loss to minimize the domain gap between the original embedding features and codebook. Experimental results show that PQM outperforms state-of-the-art deep supervised hashing and quantization methods on several image retrieval benchmarks.

Index Terms— Product Quantization, Hashing, Deep Learning

1. INTRODUCTION

Recent years have witnessed the explosion of multimedia data on the Internet with the advances of information technology. Due to high dimensional nature of multimedia data, the time and memory cost in brute-force exhaustive search become unacceptable. Therefore, ANN has become an important method for solving efficient search and indexing in multimedia systems. In the past two decades, most of the ANN works have been carried out around hashing methods [1–6]. However, quantization is also a leading approach to conduct ANN in large scale database since the metric of hashing methods are limited by the Hamming distance with less variation. In this paper, we propose a deep product quantization module inspired by PQ [7] that outperforms previous methods on several known benchmarks.

PQ and its extensions [8–10] are successful ANN methods for handling large-scale data, since PQ represents original embeddings as a concatenation of several sub-vectors. Overall, they all belong to Multi-Codebook Quantization(MCQ). For a more comprehensive survey of the product quantization and its extensions, please refer to [11]. With the revolution of deep learning, many recent research efforts have been devoted towards supervised learning of deep networks that produce compact binary codes. In MCQ under the supervised compression scenario, the ultimate optimization objective consists of three sets of variables: codebook C, backbone parameters θ and binary code B which decide quantization centers, deep features x and codebook assignment respectively.

Since discrete code B is not derivable and deep neural network is not appropriate for the assignment task, which is one of the main challenges of deep MCQ, SUBIC [12] and DPQ [13] rely on continuous relaxation of the discrete variables, namely soft quantization. However, both of them learn the binary representation implicitly and assign the codebook with ambiguous probability. DSQ solve this problem by optimizing the parameter B via Stochastic Local Search [9] which leads to computationally demanding cost. To avoid these shortcomings, inspired by straight-through estimator [14], we introduce hard quantization to deep network by directly assigning sub-vector x_m to exclusive codeword according to Euclidean distance. Compared to soft quantization, hard quantization is a more precise way of assignment which can promote the performance of retrieval directly. To the best of our knowledge, our work is the first one that adopts hard quantization without any additional parameters.

As for the optimization of codebook C, there exists huge domain gap between original embeddings and codebook in vector space while we regard codebook as the learnable parameter. The existance of this discrepancy has adverse impact on retrieval performance with Asymmetric Distance Computation (ADC). To eliminate this discrepancy, as one of our novelties, we propose a modified Mean Square Error (MSE) loss as described in Sec.2.2.2.

Overall, the main contributions of our paper can be summarized as follows: (i) We adopt hard quantization without compromising on non-derivable binary code. Notably, PQM obtains the precise binary assignment in training procedure

directly by introducing straight-through estimator. (ii) We propose a modified MSE loss to decrease the discrepancy between embeddings and the learnable codebook in the same vector space. This can decrease adverse impact of bias in quantization. (iii) PQM is a play-and-plug module for PQ, our practice of PQM combined with various backbones has achieved the state-of-the-art performance over several benchmarks.

2. PRODUCT QUANTIZATION MODULE

2.1. Architecture

The diagram of the PQM with backbone is presented on the left top of Fig. 1. All the existing supervised MCQ Deep Neural Network (DNN) architectures are composed by three modules: backbone module, quantization header which maps deep features into binary code and the supervision module for task optimization object. In particular, there are M quantization headers in PQM, and the m^{th} quantization header is responsible for m^{th} sub-vector.

With input image g, let $f(g;\theta)$ denote the backbone that embeds the input images into deep features x. The embedding x is then sliced into M sub-vectors $x = [x_1, x_2, ..., x_M]$, where each sub-vector $x_m \in \mathbb{R}^D$. Then for each sub-vector, let C_m denote corresponding learnable codeword matrix in m^{th} quantization header Q_m, which is composed by K codewords in $\mathbb{R}^D$.

Let $E(\cdot)$ be the one-hot encoder, which maps scalar into binary code $d \in \{0,1\}^K$, $s.t.$ $||d||_1 = 1$. With $E(\cdot)$, the hard code B_m can be generated by computing Euclidean distance between sub-vectors x_m and matrix C_m in quantization header Q_m:

$$B_m = E(arg\min_k ||x_m - C_m^k||_2^2). \tag{1}$$

Hence, one of optimization objective B in previous methods can be treated as intermediate variable which dependents on embedding x and codebook C without regarding it as learnable parameters. Finally, the reconstruction corresponding m^{th} sub-vector $\widetilde{x}_m$ can be represented as $C_m^{k_m^*}$ where k_m^* denotes the index of the entry in codebook C_m with the minimum approximate error. The M reconstructed vectors are concatenated to the final reconstructed vector $\widetilde{x} \in \mathbb{R}^{DM}$:

$$\begin{aligned}\widetilde{x} &= [\widetilde{x}_1, \widetilde{x}_2, ..., \widetilde{x}_m] \\ &= [C_1^{k_1^*}, C_2^{k_2^*}, ..., C_m^{k_m^*}].\end{aligned} \tag{2}$$

Since B_m is discrete and the operation $E(\cdot)$ is not a differential operation, we back-propagation gradients received by embedding vector x to the backbone network and update parameters θ directly. Following the original PQ, we can achieve the compression ratio of $\frac{32U}{Mlog_2(K)}$ when using 32-bit floating number to represent the original embedding $x \in \mathbb{R}^U$.

2.2. Loss Functions

In our experiment, loss functions can be divided into two branches. While the first branch contains supervised cross-entropy loss and center loss to optimize the embeddings x and $\widetilde{x}$ towards better retrieval performance directly; the second branch contains Mean Square Error (MSE) loss for minimizing domain gap and optimizing codebook at the same time.

2.2.1. Supervised Learning for Image Retrieval

In the task of image retrieval, cross-entropy loss can capture the requirement that the reconstructed embedding $\widetilde{x}$ need to be classified correctly. Given reconstructed vector $\widetilde{x}^i$ of sample x^i and class collection G, we can estimate the probability $p(\hat{y}^i = g|\widetilde{x}^i)$ by softmax function, where $\hat{y}^i$ is the predicted object class label for $\widetilde{x}^i$, while y^i is the ground-truth of $\widetilde{x}^i$.

Given the sample $\widetilde{x}^i$ which belongs to y^i, we can infer the incorrect classification joint probability as

$$p(\hat{y}^i \neq y^i|\widetilde{x}^i) = \prod_{g\in G, g\neq y^i} (1 - p(\hat{y}^i = g|\widetilde{x}^i)). \tag{3}$$

For the propose of pulling the embedding $\widetilde{x}^i$ from incorrect class clusters, the cross-entropy loss in PMQ can be formulated as in batch T:

$$\begin{aligned}L_{cls} = &-\frac{1}{|T|}\sum_{i=1}^{|T|} log[p(\hat{y}^i = y^i|\widetilde{x}^i)] \\ &-\frac{1}{|T|(|G|-1)}\sum_{i=1}^{|T|} log[p(\hat{y}^i \neq y^i|\widetilde{x}^i)].\end{aligned} \tag{4}$$

However, in the task of image retrieval, it is also important to encourage intra-class concentration while cross-entropy loss maintain inter-class separation. Therefore, we adopt center loss [15] to encourage reconstructed embedding has the nature of clustering in the same class.

Following [15], in the training process, we learn a center embedding $o^{y_i} \in \mathbb{R}^{MD}$ for class y_i, which can be regard as the centroid of class cluster. We find the centers by minimizing the L2 distance between embedding $\widetilde{x}^i$ and corresponding center o^{y^i}. For batch T, the optimization object is

$$L_{cen} = \sum_{i\in T} ||o^{y^i} - \widetilde{x}^i||_2^2. \tag{5}$$

In essence, with combination of modified cross-entropy loss and center loss, PQM achieves two crucial factors of metric learning: intra-class concentration and inter-class separation. Without sampling strategy such as triplet learning and pairwise learning, this is an alternative way of promoting performance in retrieval. It is worth noting that, this supervised learning imposed on reconstruction $\widetilde{x}$ directly to optimize embedding x as well assignment B. We will describe the optimization of codebook in Sec. 2.2.2.

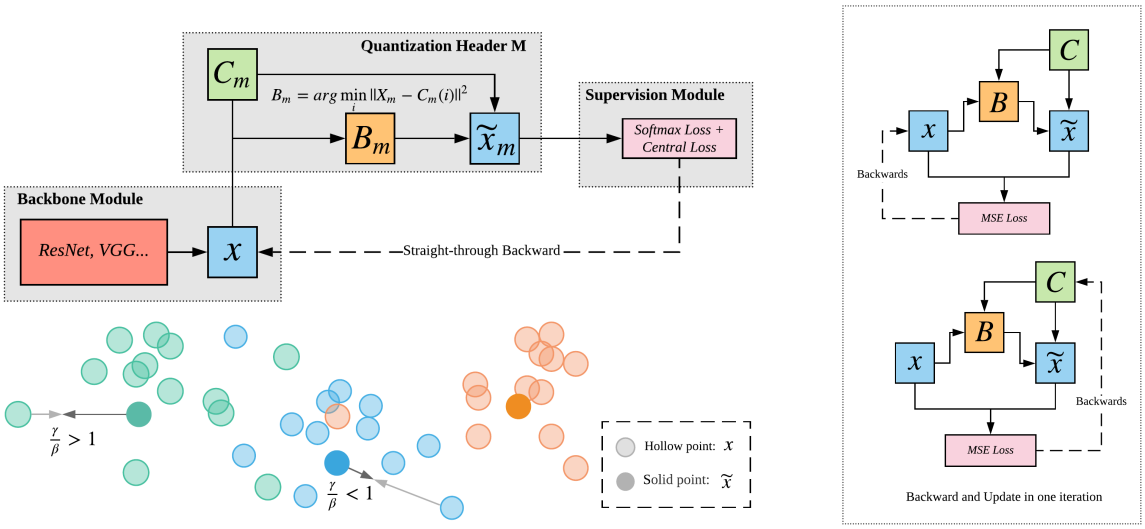

Fig. 1. Left top: the proposed PQM with backbone architecture, where x, C_m, B_m, $\widetilde{x}_m$ denote as original embedding, m^{th} codebook, m^{th} sub-vector assignment and reconstructed m^{th} sub-vector respectively. We only demonstrate m^{th} quantization header out of M quantization headers. Left bottom: Demonstration of adaption between distributions of x and $\widetilde{x}$. The length of arrow indicates the gradient scale in vector space. Green points demonstrate the situation of $\frac{\gamma}{\beta} > 1$ where the $\widetilde{x}$ has the larger gradient scale than the corresponding embedding x. Blue points demonstrate the situation of $\frac{\gamma}{\beta} < 1$ where the effect is opposite. Right: The updating strategy of C and x in one iteration via stop gradient back-propagation.

2.2.2. *Optimization of codebook*

PQM obtained discriminative codebook C by embedding both C and x in a common learnable space. We achieve this by imposing MSE operation to minimize reconstruction error in each sub-dimension:

$$MSE(x_m, \widetilde{x}_m) = ||x_m - \widetilde{x}_m||_2^2. \tag{6}$$

There are two sets of learnable parameters in PQM: the parameters of the deep network θ, determining embedding x; and the codewords in matrix C, determining $\widetilde{x}$. Instead of updating one set of parameters while fixing others in each iteration, here we optimize both C and x in one iteration by stop gradient back-propagation. This is shown in right of Fig. 1, where the assignment matrix B is detached. With $dt(\cdot)$ denoting the operation of stopping gradient back-propagation, namely $detach$ operation in PyTorch implementation, the MSE loss in PQM can be rewritten as:

$$\begin{aligned} L_{mse} = \sum_{m \in M} [\gamma MSE(dt(x_m), \widetilde{x}_m) + \\ \beta MSE(x_m, dt(\widetilde{x}_m))]. \end{aligned} \tag{7}$$

Different from original MSE which make $\widetilde{x}$ and x closer in vector space together, γ and β control the gradient scale of x and $\widetilde{x}$ respectively, as shown in left bottom of Fig. 1. If $\frac{\gamma}{\beta} > 1$, the distribution of $\widetilde{x}$ is more adaptive to the distribution of x. If $\frac{\gamma}{\beta} < 1$, we can get the opposite effect. Similar with our work, VQ-VAE [16] uses the same approach. However, VQ-VAE is targeted at image generation.

The overall loss for training the model is given by,

$$L = L_{cls} + \alpha L_{cen} + \mu L_{mse}, \tag{8}$$

where α and μ are the hyper-parameters that control the effect of each them.

In practice, we initialize the parameters of backbone by fine-tuning on the specific datasets and conduct k-means to obtain the initialized codebook.

3. EXPERIMENTS

Here, we adopt three protocols that are commonly used in previous works to capture the capability of PQM in various backbones and compare it with the state-of-the-art methods. Note that, although the search time comparison is not presented here due to space limit, all deep MCQ method techniques have the similar query time under the same distance computation technique: ADC, which has been proposed in PQ [7].

3.1. Datasets and Evaluation

Datasets. We evaluate the performance of PQM on two datasets with multi-scale: CIFAR-10 dataset consists of 60K 32×32 color images, with 6K images per class. Imagenet-100 is generated from ILSVRC 2012, which is a dataset with over 1.2 million images covering 1K categories.

Evaluation. In each protocol, we evaluate our model by employing the mean Average Precision (mAP) metric, a

	Bit	ITQ	SQ	CNNH	DSH	HashNet	PQ+DF	DQN	SUBIC	DTQ	DSQ	DPQ	**Ours**
Prot. 1	16	0.241	0.621	0.537	0.619	0.686	0.709	0.598	0.656	0.704	**0.721**	-	0.717
	32	0.243	0.644	0.542	0.657	0.692	0.715	0.610	0.679	0.719	0.735	-	**0.738**
	64	0.253	0.655	0.577	0.671	0.719	0.718	0.610	0.685	0.732	0.742	-	**0.753**
Prot. 2	12	0.108	0.541	0.543	0.616	0.648	0.606	0.554	0.635	0.684	-	0.741	**0.744**
	24	0.109	0.586	0.560	0.651	0.673	0.623	0.558	0.672	0.693	-	**0.754**	0.745
	36	0.112	0.596	0.564	0.661	0.675	0.624	0.564	0.682	0.722	-	0.754	**0.755**
	48	0.118	0.607	0.557	0.675	0.676	0.601	0.580	0.686	0.728	-	0.754	**0.760**

Table 1. Retrieval performance (mAP) on the CIFAR-10 dataset with AlexNet and 3CNet according to Protocol 1 (Prot.1) and Protocol 2 (Prot.2) respectively. Missing results that were not reported and are expected not to be competitive.

Method	16-bit	32-bit	64-bit
LSH	0.101	0.235	0.360
ITQ	0.323	0.462	0.552
DHN	0.311	0.472	0.573
HashNet	0.506	0.631	0.684
DBR-v3	0.733	0.761	0.769
HDT	0.838	0.822	0.812
DPQ	**0.886**	0.877	0.866
Ours	0.844	**0.879**	**0.874**

Table 2. Retrieval performance (mAP) on the Imagenet-100 dataset with ResNet-50 and varying number of bits according to Protocol 3.

universal metric in image retrieval. We report the results of mAP@all and mAP@1000 for CIFAR-10 and Imagenet-100 respectively.

3.2. Results

Methods. We compare PQM with a wide range of supervised compact coding methods including binary hashing methods: ITQ [17], SQ [18], CNNH [2], DSH [3], HashNet [1], DQN [5], SUBIC [12], DTQ [19], DSQ [20], DPQ [13], LSH [4], DHN [21], DBR-v3 [22], HDT [6]. We evaluate HashNet and DTQ with 3CNet by using the implementation generously provided by the authors. Moreover, we evaluate traditional PQ with the deep features as input, namely PQ+DF. We extract the deep features from corresponding backbone which has been pre-trained with the supervision towards classification. We use the PQ implementation from the search libaray FAISS [23] to evaluate quantization part in PQ+DF. Other results for previous methods were obtained directly from [13], [20] and [3].

3.2.1. CIFAR-10 on Protocol 1 and Protocol 2

Protocol 1: CIFAR-10 with 3CNet. In this protocol, we follow the [13]. The 3CNet in this protocol is the same as architecture proposed by DSH. As for the hyper-parameters M and K, we split the original embeddings into $M = \{4, 6, 6, 8\}$ partitions, meanwhile $K = \{8, 16, 64, 64\}$, such that we vary bit-rates in $\{12, 24, 36, 48\}$. To be fair, the end of backbone 3CNet have $U = 120$ units, so that each codeword in codebook is 30-dimensional. We achieve this by adding a fully connected layer, on top of $fc1$ in DSH.

Protocol 2: CIFAR-10 with AlexNet. In this protocol, we follow the settings in [20]. We adopt the same core architecture (AlexNet) and generate original embedding $x \in \mathbb{R}^{256}$ by adding a fully connected layer on top of $fc2$ in classfier of AlexNet. We vary $M = \{2, 4, 8\}$ to obtain various bit-rates in $\{16, 32, 64\}$ while $K = 64$ is fixed.

As shown in Table 1, our PQM method achieves state-of-the-art results in most of the bit-rates. In particular, we evaluate the baseline PQ+DF. This baseline has been largely ignored in previous works. However, it achieves better results than several previous methods which impose pertinent supervised signal towards quantization or hashing.

3.2.2. Imagenet-100 on Protocol 3

Protocol 3: Imagenet-100 with ResNet50. In this protocol, we follow the settings in [1]. By adding a fully connected layer on the top of the last $avgpool$ in ResNet50, we generate 512-dimensional embeddings. As suggested by [13], we fix $M = 8$ while varying K among $\{4, 16, 256\}$ achieve bit-rates $\{16, 32, 64\}$. As shown in Table. 2, ours method also achieves state-of-the-art results on high bit-rates.

4. CONCLUSION

In this paper, we propose a deep product quantization module for efficient and fast image retrieval. By learning discriminative codebook and hard assignment jointly, we achieve a more precise way of MCQ compared to soft quantization in DNN. And we have shown the superiority of hard quantization in Sec.3. We show that, combined with various backbones, PQM achieves state-of-the-art results on multiple protocols that are commonly used in previous works.

5. ACKNOWLEDGEMENT

This work was supported by National Natural Science Foundation of China under Grant U1611461.

6. REFERENCES

[1] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Philip S Yu, "Hashnet: Deep learning to hash by continuation," in *ICCV*, 2017, pp. 5608–5617.

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

[3] Haomiao Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen, "Deep supervised hashing for fast image retrieval," in *CVPR*, 2016, pp. 2064–2072.

[4] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al., "Similarity search in high dimensions via hashing," in *Vldb*, 1999, vol. 99, pp. 518–529.

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

[6] Martin Loncaric, Bowei Liu, and Ryan Weber, "Learning hash codes via hamming distance targets," *arXiv preprint arXiv:1810.01008*, 2018.

[7] Herve Jegou, Matthijs Douze, and Cordelia Schmid, "Product quantization for nearest neighbor search," *PAMI*, vol. 33, no. 1, pp. 117–128, 2010.

[8] Artem Babenko and Victor Lempitsky, "Additive quantization for extreme vector compression," in *CVPR*, 2014, pp. 931–938.

[9] Julieta Martinez, Joris Clement, Holger H Hoos, and James J Little, "Revisiting additive quantization," in *ECCV*. Springer, 2016, pp. 137–153.

[10] Ting Zhang, Chao Du, and Jingdong Wang, "Composite quantization for approximate nearest neighbor search.," in *ICML*, 2014, vol. 2, p. 3.

[11] Yusuke Matsui, Yusuke Uchida, Hervé Jégou, and Shin'ichi Satoh, "A survey of product quantization," *ITE Transactions on Media Technology and Applications*, vol. 6, no. 1, pp. 2–10, 2018.

[12] Himalaya Jain, Joaquin Zepeda, Patrick Pérez, and Rémi Gribonval, "Subic: A supervised, structured binary code for image search," in *ICCV*, 2017, pp. 833–842.

[13] Benjamin Klein and Lior Wolf, "End-to-end supervised product quantization for image search and retrieval," in *CVPR*, 2019, pp. 5041–5050.

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

[15] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao, "A discriminative feature learning approach for deep face recognition," in *ECCV*. Springer, 2016, pp. 499–515.

[16] Aaron van den Oord, Oriol Vinyals, et al., "Neural discrete representation learning," in *NeurIPS*, 2017, pp. 6306–6315.

[17] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," *PAMI*, vol. 35, no. 12, pp. 2916–2929, 2012.

[18] Xiaojuan Wang, Ting Zhang, Guo-Jun Qi, Jinhui Tang, and Jingdong Wang, "Supervised quantization for similarity search," in *CVPR*, 2016, pp. 2018–2026.

[19] Bin Liu, Yue Cao, Mingsheng Long, Jianmin Wang, and Jingdong Wang, "Deep triplet quantization," in *MM*. ACM, 2018, pp. 755–763.

[20] Sepehr Eghbali and Ladan Tahvildari, "Deep spherical quantization for image search," in *CVPR*, 2019, pp. 11690–11699.

[21] Han Zhu, Mingsheng Long, Jianmin Wang, and Yue Cao, "Deep hashing network for efficient similarity retrieval," in *AAAI*, 2016.

[22] Xuchao Lu, Li Song, Rong Xie, Xiaokang Yang, and Wenjun Zhang, "Deep binary representation for efficient image retrieval," *Advances in Multimedia*, vol. 2017, 2017.

[23] Jeff Johnson, Matthijs Douze, and Hervé Jégou, "Billion-scale similarity search with gpus," *arXiv preprint arXiv:1702.08734*, 2017.