The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Distributed Composite Quantization

Weixiang Hong, Jingjing Meng, Junsong Yuan
School of Electrical and Electronic Engineering,
Nanyang Technological University, Singapore
{wxhong,jingjing.meng,JSYUAN}@ntu.edu.sg

Abstract

Approximate nearest neighbor (ANN) search is a fundamental problem in computer vision, machine learning and information retrieval. Recently, quantization-based methods have drawn a lot of attention due to their superior accuracy and comparable efficiency compared with traditional hashing techniques. However, despite the prosperity of quantization techniques, they are all designed for the centralized setting, *i.e.*, quantization is performed on the data on a single machine. This makes it difficult to scale these techniques to large-scale datasets. Built upon the Composite Quantization, we propose a novel quantization algorithm for data distributed across different nodes of an arbitrary network. The proposed Distributed Composite Quantization (DCQ) decomposes Composite Quantization into a set of decentralized subproblems such that each node solves its own sub-problem on its local data, meanwhile is still able to attain consistent quantizers thanks to the consensus constraint. Since there is no exchange of training data across the nodes in the learning process, the communication cost of our method is low. Extensive experiments on ANN search and image retrieval tasks validate that the proposed DCQ significantly improves Composite Quantization in both efficiency and scale, while still maintaining competitive accuracy.

1. Introduction

Nearest neighbor (NN) search has wide applications in computer vision, machine learning and information retrieval, *e.g.*, image retrieval, object instance search and k-NN classifier (Shakhnarovich, Indyk, and Darrell 2006), *etc.* The straightforward solution, linear scan, is both computationally and memory intensive in large scale high-dimensional cases, thus is not preferable in practice. Therefore, there have been a lot of interests in algorithms that perform approximate nearest neighbor (ANN) search.

ANN search is traditionally addressed with hashing methods (Wang et al. 2016b; Hong, Yuan, and Das Bhattacharjee 2017; Liu et al. 2017; Hong, Meng, and Yuan 2018), which have been comprehensively surveyed in (Wang et al. 2017). However, a family of methods based on vector quantization (Jegou, Douze, and Schmid 2011; Babenko and Lempitsky 2015; Heo, Lin, and Yoon 2014; Zhang et al. 2015; Wang et al. 2016a; Zhang and Wang 2016) has recently triggered interests from computer vision, machine learning and multimedia retrieval communities, due to its superior accuracy and comparable efficiency compared with hashing techniques. Different from hashing, these methods perform ANN search by learning numerous *decimal quantizers*, hence do not suffer from the quantization loss from decimal space to binary space, which is the main reason of their better accuracy compared with hashing. Meanwhile, quantization-based methods are also efficient thanks to the smart use of lookup table.

Despite the prosperity of the quantization techniques, they are all designed for the centralized setting, or in other words, are single-machine approaches. Nevertheless, due to the explosion in size and complexity of modern datasets, more and more real-world applications need to deal with data distributed across different locations, such as distributed databases (Corbett et al. 2013), images/videos over the networks (Bhattacharjee et al. 2016; Meng et al. 2016), *etc.* Furthermore, in some applications, the data is inherently distributed. For example, in video surveillance (Cong, Yuan, and Liu 2011) and sensor networks (Liang et al. 2014), the data is collected at distributed sites. In such contexts, the quantizers should be learned based on the entire dataset in order to get unbiased quantization codes for the data. One intuitive way is gathering all data together at a central server before training, but it is not a feasible option because of the huge communication overhead. Besides, directly training on large-scale data is often prohibitive in both time and space, which further prevents it from practical applications. As a consequence, it is important to develop quantization algorithms that are both powerful enough to capture the complexity of large scale data, and scalable enough to process huge datasets in parallel. However, to our knowledge, this critical and challenging problem has never been explored in the literature.

This paper proposes Distributed Composite Quantization (DCQ) for data which is distributed across different nodes of an arbitrary network (*e.g.*, Figure 1). Unlike the conventional centralized methods which require gathering the distributed data from all nodes to learn common quantizers, our method learns such quantizers in a distributed manner. Each node learns a set of dictionaries on its local data, and only exchanges the local dictionaries with other nodes

(See Section 4.4.2 for details). To this end, we decompose a centralized quantization model into a set of decentralized sub-problems with consensus constraints and the alternating direction method of multipliers (ADMM) (Hestenes 1969; Powell and Authority 1967). We prove the convergence of ADMM for our non-convex Lagrangian by reformulating it as a global consensus problem with sufficiently large penalty parameter (Hong, Luo, and Razaviyayn 2016), as a result, these sub-problems can be efficiently solved in parallel within a few iterations, and all the nodes obtain consistent quantizers learned from the distributed data. The main contributions of this paper can be summarized as follows:

- We raise a challenging problem in quantization, *i.e.*, quantization of distributed data. This problem is essential for making quantization techniques applicable for real-world large-scale retrieval systems. However, to our best knowledge, it has never been discussed in any literature.
- We propose Distributed Composite Quantization (DCQ) to perform quantization on distributed data. Since there is no exchange of training data across the nodes in the learning process, the communication cost of our method is low. Moreover, our approach can adapt to arbitrary network topologies.
- We prove the convergence of ADMM for the non-convex Lagrangian involved in our DCQ. Although the theoretical convergence property of the ADMM update rule for the general non-convex problem is still an open question (Bertsekas 1989), ADMM has been proved to converge for a family of non-convex problems under certain assumptions (Hong, Luo, and Razaviyayn 2016). We prove that the Lagrangian of our proposed DCQ satisfies those required assumptions, as a result, the theoretical convergence of DCQ is guaranteed with a sufficiently large penalty parameter.
- We conduct extensive experimental evaluations on approximate nearest neighbor search and image retrieval in a distributed setting. Experimental results validate that the proposed method can improve Composite Quantization in both efficiency and scale, while still maintaining competitive accuracy.

2. Related Work

Vector quantization algorithms can be coarsely classified into two categories: (1) partition-based quantization such as PQ (Jegou, Douze, and Schmid 2011), OPQ (Ge et al. 2013), and CKM (Norouzi and Fleet 2013) (2) addition-based quantization such as AQ (Babenko and Lempitsky 2014), CQ (Zhang, Du, and Wang 2014), and LSQ (Martinez et al. 2016). Partition-based methods usually devide the data space into a number of disjoint subspaces and quantizes each subspace separately, while addition-based approaches typically approximate a database vector using the addition of dictionary words selected from different dictionaries.

Generally speaking, addition-based quantization methods perform better than partition-based ones, because addition-based quantizations do not decompose data space into orthogonal subspaces and thus make no subspace independence assumptions. Our approach also goes in this direction.

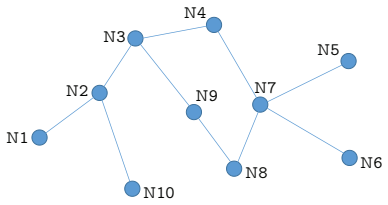


Figure 1: A randomly generated network with 10 nodes. Such a network can be modeled with an undirected and connected graph.

To be more specific, we generalize Composite Quantization (CQ) to the distributed scenarios.

3. Problem Formulation

We start by introducing the Composite Quantization that works in centralized setteings, then we will extend it to the distributed settings.

3.1. Composite Quantization

Composite Quantization aims to approximate a vector $\mathrm{x} \in \mathbb{R}^d$ by the composition of M vectors $\{\mathrm{C}_{1k_1}, \mathrm{C}_{2k_2}, \ldots, \mathrm{C}_{Mk_M}\}$, each of which is selected from a dictionary with K elements, *i.e.*, C_{mk_m} is the k_mth element in the dictionary C_m, and $\forall m \in \{1, 2, \ldots, M\}$, $\mathrm{C}_m = \{\mathrm{C}_{m1}, \mathrm{C}_{m2}, \ldots, \mathrm{C}_{mK}\}$.

Let $\bar{\mathrm{x}} = \sum_{m=1}^{M} \mathrm{C}_{mk_m}$ be the approximation of vector x. The accuracy of nearest neighbor search relies on the quality of the distance approximation, *i.e.*, how small is the difference between the distance of the query q to the vector x and the distance to the approximation $\bar{\mathrm{x}}$. According to the triangle inequality, $\big|\,||\mathrm{q}-\mathrm{x}||_2 - ||\mathrm{q}-\bar{\mathrm{x}}||_2\big| \leq ||\mathrm{x}-\bar{\mathrm{x}}||_2$, the distance approximation error in ANN search is bounded by the vector approximation error, which is formulated as follows,

$$\min_{\mathrm{C}_{mk_m}} \sum_{\mathrm{x}\in\mathrm{X}} ||\mathrm{x} - \sum_{m=1}^{M} \mathrm{C}_{mk_m}||_2^2, \tag{1}$$

where C_{mk_m} is the selected element from the dictionary C_m for the database vector x.

With the approximation $\bar{\mathrm{x}} = \sum_{m=1}^{M} \mathrm{C}_{mk_m}$, we have

$$\begin{aligned} ||\mathrm{q}-\bar{\mathrm{x}}||_2^2 = &\sum_{m=1}^{M} ||\mathrm{q} - \mathrm{C}_{mk_m}||_2^2 \\ &- (M-1)||q||_2^2 + \sum_{i=1}^{M}\sum_{j=1,j\neq i}^{M} \mathrm{C}_{ik_i}^T \mathrm{C}_{jk_j}. \end{aligned} \tag{2}$$

Given the query q, the first term can be efficiently computed using lookup table, and the second term is constant for all database vectors, hence unnecessary to compute for ANN search. For the third term, we constrain it to be a constant ϵ, *i.e.*, $\sum_{i=1}^{M}\sum_{j=1,j\neq i}^{M} \mathrm{C}_{ik_i}^T \mathrm{C}_{jk_j} = \epsilon$, which is referred as constant inter-dictionary-element-product in Composite Quantization.

Let $\mathrm{C} = [\mathrm{C}_1\mathrm{C}_2\cdots\mathrm{C}_M] \in \mathbb{R}^{d\times MK}$ be the whole dictionary, $\mathrm{B} = [\mathrm{b}_1^T\mathrm{b}_2^T\cdots\mathrm{b}_N^T] \in \mathbb{R}^{MK\times N}$ be the matrix consisting of all codes. Furthermore, we impose three extra constraints on B to fully complete the formulation, namely, $\mathrm{b}_n = [\mathrm{b}_{n1}\mathrm{b}_{n2}\cdots\mathrm{b}_{nM}] \in \{0,1\}^{MK}$, $\mathrm{b}_{nm} \in \{0,1\}^K$ and $||\mathrm{b}_{nm}||_1 = 1$. Finally, the optimization problem is formulated as:

$$\begin{aligned} \min_{\mathrm{C},\mathrm{B},\epsilon} \quad & ||\mathrm{X}-\mathrm{CB}||_{\mathrm{F}}^2 \\ \text{s.t.} \quad & \sum_{i=1}^{M}\sum_{j=1,j\neq i}^{M} \mathrm{b}_{ni}^T\mathrm{C}_i^T\mathrm{C}_j\mathrm{b}_{nj} = \epsilon, \forall n = 1,2,\ldots,N. \end{aligned} \tag{3}$$

By adopting the quadratic penalty method, Composite Quantization relaxed the optimization problem in Equation (3) as:

$$\phi(\mathrm{C},\mathrm{B},\epsilon) = ||\mathrm{X}-\mathrm{CB}||_{\mathrm{F}}^2 + \mu\sum_{n=1}^{N}(\sum_{i\neq j}^{M}\mathrm{b}_{ni}^T\mathrm{C}_i^T\mathrm{C}_j\mathrm{b}_{nj}-\epsilon)^2, \tag{4}$$

where $\sum_{i\neq j}^{M} = \sum_{i=1}^{M}\sum_{j=1,j\neq i}^{M}$.

3.2. Distributed Composite Quantization

Next, we generalize Composite Quantization to the distributed scenarios. In our setting, the data is distributed across a set of P nodes in a network (*e.g.*, Figure 1). On the s-th node, there is a local set of N^s data points, denoted in matrix form as X^s. The global data $\mathrm{X} = \cup_{s=1}^{P}\mathrm{X}^s$ is then a concatenation of the local data matrix. When the data is distributed across the P nodes in an arbitrary network, the objective in Equation (4) can be rewritten as:

$$\begin{aligned} \phi(\mathrm{C},\mathrm{B},\epsilon) = &\sum_{s=1}^{P}||\mathrm{X}^s-\mathrm{CB}^s||_{\mathrm{F}}^2 \\ &+\mu\sum_{s=1}^{P}\sum_{n=1}^{N^s}(\sum_{i\neq j}^{M}{\mathrm{b}_{ni}^s}^T\mathrm{C}_i^T\mathrm{C}_j\mathrm{b}_{nj}^s-\epsilon)^2, \end{aligned} \tag{5}$$

where B^s denotes the composition codes that belongs to the s-th node. Throughout this paper, we use superscript to denote the identity number of each node. We will further use C^s and ϵ^s to denote the local copies of C and ϵ on the s-th node.

4. Optimization

In this section, we present a distributed optimization algorithm to learn composite quantization in a decentralized scenario. The problem formulated in Equation (5) is a mixed-binary-integer program, we use the iterative optimization technique to iteratively solve it. Each iteration alternatively updates the composition matrix B, constant inter-dictionary-element-product constraint ϵ and dictionary C.

4.1. Update B

Each B^s can be locally updated in parallel on each node. For example, to update b_n^s, the codes for the n-th data vector on the s-th node, the subproblem to tackle is as following:

$$\phi(\mathrm{b}_n^s) = ||\mathrm{X}_n^s-\mathrm{C}^s\mathrm{b}_n^s||_2^2 + \mu(\sum_{i\neq j}^{M}{\mathrm{b}_{ni}^s}^T\mathrm{C}_i^{sT}\mathrm{C}_j^s\mathrm{b}_{nj}^s-\epsilon^s)^2, \tag{6}$$

where X_n^s denotes the n-th vector on the s-th node. Also there are three extra constraints for Equation (6): $\mathrm{b}_n^s = [\mathrm{b}_{n1}^s\mathrm{b}_{n2}^s\cdots\mathrm{b}_{nM}^s] \in \{0,1\}^{MK}$, $\mathrm{b}_{nm}^s \in \{0,1\}^K$ and $||\mathrm{b}_{nm}^s||_1 = 1$. The problem is essentially a high-order MRF problem and NP-hard. To efficiently find a feasible local optimum, we again adopt the alternative optimization technique to solve the M subvectors $\{\mathrm{b}_{nm}^s\}$ in turn. Specifically, given $\{\mathrm{b}_{nl}^s\}_{l\neq m}$ fixed, we exhaustively check all the elements in the dictionary C_m^s, and find the element that minimizes the objective function value in Equation 6, then update $\{\mathrm{b}_{nm}^s\}$ by setting the corresponding entry in b_{nm}^s to be 1 and all the others to be 0.

4.2. Update ϵ

Given fixed C and B in Equation (5), the objective function turns out to be quadratic with respect to ϵ. Under the distributed setting of this work, the optimal solution to ϵ can be solved in a three-step fashion. First, we update each local copy ϵ^s of ϵ as following:

$$\epsilon^s = \frac{1}{N^s}\sum_{n=1}^{N^s}\sum_{i\neq j}^{M}{\mathrm{b}_{ni}^s}^T\mathrm{C}_i^{sT}\mathrm{C}_j^s\mathrm{b}_{nj}^s. \tag{7}$$

Then we can select any node as the coordinator to conduct the following computation:

$$\epsilon = \frac{1}{N}\sum_{s=1}^{P}N^s\epsilon^s. \tag{8}$$

Finally, the selected coordinator shall boardcast the new ϵ to each nodes to replace the previous local copies $\{\epsilon^s\}$.

4.3. Update C

When B and ϵ are fixed, the dictionary C is shared across all nodes, which makes the problem hard to tackle. In order to make the objective separable, we enforce the consensus constraints $\mathrm{C}^s = \mathrm{C}^t$, for $\forall s,t \in \{1,2,\ldots,P\}$ for $\{\mathrm{C}^s\}$ on all nodes, thus, we can transform Equation (5) to the following form without introducing any relaxation.

$$\begin{aligned} \min_{\{\mathrm{C}^s\}} \quad & \sum_{s=1}^{P}||\mathrm{X}^s-\mathrm{C}^s\mathrm{B}^s||_{\mathrm{F}}^2 \\ & +\mu\sum_{s=1}^{P}\sum_{n=1}^{N_s}(\sum_{i\neq j}^{M}{\mathrm{b}_{ni}^s}^T\mathrm{C}_i^{sT}\mathrm{C}_j^s\mathrm{b}_{nj}^s-\epsilon^s)^2 \\ \text{s.t.} \quad & \mathrm{C}^s = \mathrm{C}^t, \forall s,t \in \{1,2,\ldots,P\}. \end{aligned} \tag{9}$$

The consensus constraint implies that all the local dictionaries should be consistent. In this way, the additive objective in Equation (5) is split into a set of separable objectives. Thanks to the transitivity between neighboring nodes in a connected graph, we are allowed to consider only the

constraints between the neighboring nodes rather than all the constraints. For example, if the consensus constraints between all neighboring nodes are satisfied in Figure 1, then $\mathrm{C}^3 = \mathrm{C}^6$ is naturally satisfied due to the fact that $\mathrm{C}^3 = \mathrm{C}^4 = \mathrm{C}^7 = \mathrm{C}^6$. Based on this observation, Equation (9) can be equivalently reformulated as:

$$\begin{aligned} \min_{\{\mathrm{C}^s\}} \quad & \sum_{s=1}^{P} ||\mathrm{X}^s - \mathrm{C}^s\mathrm{B}^s||_{\mathrm{F}}^2 \\ & + \mu \sum_{s=1}^{P} \sum_{n=1}^{N^s} (\sum_{i \neq j}^{M} {\mathrm{b}_{ni}^s}^T \mathrm{C}_i^{sT} \mathrm{C}_j^s \mathrm{b}_{nj}^s - \epsilon^s)^2 \\ \text{s.t.} \quad & \mathrm{C}^s = \mathrm{C}^{s'}, s' \in \mathcal{N}(s), \forall s \in \{1, 2, \ldots, P\}. \end{aligned} \tag{10}$$

where $\mathcal{N}(s)$ represents the neighbors of the s-th node.

Next we show how the alternating direction method of multipliers (ADMM) (Zhang and Kwok 2014; Leng et al. 2015; Liu et al. 2016) can be applied to decompose the global problem (10) into several local subproblems.

4.3.1 Distributed Learning ADMM is a variant of the augmented Lagrangian scheme that blends the decomposability of dual ascent with the method of multipliers. For our specific problem (10), the augmented Lagrangian is:

$$\begin{aligned} L(\mathrm{C}^s, \Lambda^{s,s'}) = & \sum_{s=1}^{P} ||\mathrm{X}^s - \mathrm{C}^s\mathrm{B}^s||_{\mathrm{F}}^2 \\ & + \mu \sum_{s=1}^{P} \sum_{n=1}^{N_s} (\sum_{i \neq j}^{M} {\mathrm{b}_{ni}^s}^T \mathrm{C}_i^{sT} \mathrm{C}_j^s \mathrm{b}_{nj}^s - \epsilon^s)^2 \\ & + \sum_{s=1}^{P} \sum_{s' \in \mathcal{N}(s)} \mathrm{tr}\big({\Lambda^{s,s'}}^T (\mathrm{C}^s - \mathrm{C}^{s'})\big) \\ & + \frac{\rho}{2} \sum_{s=1}^{P} \sum_{s' \in \mathcal{N}(s)} ||\mathrm{C}^s - \mathrm{C}^{s'}||_{\mathrm{F}}^2, \end{aligned} \tag{11}$$

where $\Lambda^{s,s'}$ is the Lagrangian multipliers corresponding to the constraints $\mathrm{C}^s = \mathrm{C}^{s'}, \forall s \in \{1, 2, \ldots, P\}$ and $s' \in \mathcal{N}(s)$. $\rho > 0$ is the penalty parameter of augmented Lagrangian. ADMM solves a problem of this form by repeating the following two steps (Liang et al. 2014):

$$\begin{aligned} \mathrm{C}^s \coloneqq & \arg\min_{\mathrm{C}^s} \sum_{s=1}^{P} ||\mathrm{X}^s - \mathrm{C}^s\mathrm{B}^s||_{\mathrm{F}}^2 \\ & + \mu \sum_{s=1}^{P} \sum_{n=1}^{N_s} (\sum_{i \neq j}^{M} {\mathrm{b}_{ni}^s}^T \mathrm{C}_i^{sT} \mathrm{C}_j^s \mathrm{b}_{nj}^s - \epsilon^s)^2 \\ & + \sum_{s=1}^{P} \sum_{s' \in \mathcal{N}(s)} \mathrm{tr}\big({\Lambda^{s,s'}}^T (\mathrm{C}^s - \mathrm{C}^{s'})\big) \\ & + \frac{\rho}{2} \sum_{s=1}^{P} \sum_{s' \in \mathcal{N}(s)} ||\mathrm{C}^s - \mathrm{C}^{s'}||_{\mathrm{F}}^2 \end{aligned} \tag{12a}$$

$$\Lambda^{s,s'} \coloneqq \Lambda^{s,s'} + \rho(\mathrm{C}^s - \mathrm{C}^{s'}). \tag{12b}$$

Despite the algorithm's elegance in form, the subproblems are still difficult to solve.

4.3.2 Simplification of Lagrange Multipliers Due to the symmetry of an undirected graph, it is clear that if $s' \in \mathcal{N}(s)$ then $s \in \mathcal{N}(s')$. That is to say, every available constraint in the Equation (10) has been considered at least twice, *i.e.*, $\mathrm{C}^s = \mathrm{C}^{s'}$ and $\mathrm{C}^{s'} = \mathrm{C}^s$, which suggests that we can simplify the update rules in Equation (12). First of all, we can rewrite the third term of Equation (12a) in another form as:

$$\begin{aligned} & \sum_{s=1}^{P} \sum_{s' \in \mathcal{N}(s)} \mathrm{tr}\big({\Lambda^{s,s'}}^T (\mathrm{C}^s - \mathrm{C}^{s'})\big) \\ = & \sum_{s=1}^{P} \sum_{s' \in \mathcal{N}(s)} \mathrm{tr}\big({\Lambda^{s,s'}}^T \mathrm{C}^s\big) - \sum_{s=1}^{P} \sum_{s' \in \mathcal{N}(s)} \mathrm{tr}\big({\Lambda^{s,s'}}^T \mathrm{C}^{s'}\big) \\ = & \sum_{s=1}^{P} \sum_{s' \in \mathcal{N}(s)} \mathrm{tr}\big({\Lambda^{s,s'}}^T \mathrm{C}^s\big) - \sum_{s=1}^{P} \sum_{s' \in \mathcal{N}(s)} \mathrm{tr}\big({\Lambda^{s',s}}^T \mathrm{C}^s\big) \\ = & \sum_{s=1}^{P} \mathrm{tr}\big(\sum_{s' \in \mathcal{N}(s)} (\Lambda^{s,s'} - \Lambda^{s',s})^T \mathrm{C}^s\big). \end{aligned} \tag{13}$$

In addition, owing to the symmetric characteristics, we can easily write down the symmetrical counterpart of Equation (12b) as following:

$$\Lambda^{s',s} \coloneqq \Lambda^{s',s} + \rho\big(\mathrm{C}^{s'} - \mathrm{C}^s\big). \tag{14}$$

For any two adjacent nodes, with Equation (12b) and (14), we have:

$$\Lambda^{s,s'} - \Lambda^{s',s} \coloneqq \big(\Lambda^{s,s'} - \Lambda^{s',s}\big) + 2\rho\big(\mathrm{C}^s - \mathrm{C}^{s'}\big). \tag{15}$$

Therefore, by defining P new Lagrange Multipliers Λ^s as:

$$\Lambda^s = \sum_{s' \in \mathcal{N}(s)} (\Lambda^{s,s'} - \Lambda^{s',s}). \tag{16}$$

The update rule of ADMM in Equation (12) can be simplified as:

$$\begin{aligned} \mathrm{C}^s \coloneqq & \arg\min_{\mathrm{C}^s} \sum_{s=1}^{P} ||\mathrm{X}^s - \mathrm{C}^s\mathrm{B}^s||_{\mathrm{F}}^2 \\ & + \mu \sum_{s=1}^{P} \sum_{n=1}^{N_s} (\sum_{i \neq j}^{M} {\mathrm{b}_{ni}^s}^T \mathrm{C}_i^{sT} \mathrm{C}_j^s \mathrm{b}_{nj}^s - \epsilon^s)^2 \\ & + \sum_{s=1}^{P} \mathrm{tr}\big({\Lambda^s}^T \mathrm{C}^s\big) \\ & + \frac{\rho}{2} \sum_{s=1}^{P} \sum_{s' \in \mathcal{N}(s)} ||\mathrm{C}^s - \mathrm{C}^{s'}||_{\mathrm{F}}^2 \end{aligned} \tag{17a}$$

$$\Lambda^s \coloneqq \Lambda^s + 2\rho(\mathrm{C}^s - \mathrm{C}^{s'}). \tag{17b}$$

Obviously, the updates of local variables C^s and Λ^s in Equation (17) can be separated into P subproblems, and thus can be carried out independently in parallel across the nodes.

4.3.3 Solution to subproblems At last, we show how to solve the subproblem on each node. Formally, the s-th subproblem on the s-th node can be written as following:

$$\begin{aligned}\phi(\mathrm{C}^s) = &\sum_{n=1}^{N_s} ||\mathrm{X}_n^s - \mathrm{C}^s\mathrm{b}_n^s||_\mathrm{F}^2 \\ &+ \mu \sum_{n=1}^{N_s} (\sum_{i\neq j}^{M} {\mathrm{b}_{ni}^s}^T {\mathrm{C}_i^s}^T \mathrm{C}_j^s \mathrm{b}_{nj}^s - \epsilon^s)^2 \\ &+ \mathrm{tr}\big({\Lambda^s}^T \mathrm{C}^s\big) + \frac{\rho}{2} \sum_{s'\in\mathcal{N}(s)} ||\mathrm{C}^s - \mathrm{C}^{s'}||_\mathrm{F}^2,\end{aligned} \tag{18}$$

which is an unconstrained nonlinear optimization problem with respect to C^s. Following Composite Quantization (Zhang, Du, and Wang 2014), we solve it with the L-BFGS algorithm (Nocedal 1980; Liu and Nocedal 1989), the limited-memory version of the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm. The L-BFGS method approximates the full Hessian matrix using a few vectors, thus only needs the function value and function gradient as its inputs. We use the publicly available implementation of L-BFGS [1]. The partial derivative with respect to C_m^s is as follows:

$$\begin{aligned}\frac{\partial\phi(\mathrm{C}^s)}{\partial \mathrm{C}_m^s} = &\sum_{n=1}^{N_s} \big[2(\sum_{l=1}^{M} \mathrm{C}_l^s \mathrm{b}_{nl}^s - \mathrm{X}_n^s){\mathrm{b}_{nm}^s}^T \\ &+ 4\mu(\sum_{i\neq j}^{M} {\mathrm{b}_{ni}^s}^T {\mathrm{C}_i^s}^T \mathrm{C}_j^s \mathrm{b}_{nj}^s - \epsilon^s)(\sum_{l\neq m}^{M} \mathrm{C}_l^s \mathrm{b}_{nl}){\mathrm{b}_{nm}^s}^T\big] \\ &+ \Lambda^s + \rho \sum_{s'\in\mathcal{N}(s)} (\mathrm{C}_m^s - \mathrm{C}_m^{s'}).\end{aligned} \tag{19}$$

Equation (18) and (19) will serve as the inputs for the L-BFGS solver.

4.4. Analysis

We have presented the whole procedure of our proposed DCQ. Note that the update of local C^s, B^s, ϵ^s and Λ^s can all be conducted in parallel on each node, which is the key factor for our method to work in a distributed setting.

4.4.1 Analysis on Convergence It is easy to verify that Equation (5) is lower-bounded (not smaller than 0), and the updates for B and ϵ always decrease the value of Equation (5), hence, the convergence of DCQ is guaranteed if one can prove the convergence of ADMM for Equation (11), which is non-convex with respect to C.

For the general non-convex problem, the theoretical convergence property of the ADMM is still an open question (Bertsekas 1989), nevertheless, ADMM has been proved to converge for a family of non-convex problems under certain assumptions (Hong, Luo, and Razaviyayn 2016). Luckily, our problem Equation (11) satisfies those required assumptions, as a result, the theoretical convergence of ADMM for Equation (11) can be guaranteed with a sufficiently large ρ. The detailed proof is included in the supplementary material.

[1] http://www.chokkan.org/software/liblbfgs/

4.4.2 Analysis on Communication Complexity Here we analyze the communication complexity of the proposed distributed composite quantization. Recall that M denotes the number of dictionaries, K denotes the number of elements in each dictionary, d denotes the dimension of data, and N^s is the number of local samples in the s-th node.

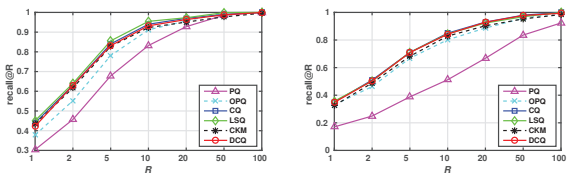
In our algorithm, each node shares the dictionary C^s with its neighboring nodes, and shares N^s as well as the constant constraint ϵ with the coordinator. Supposing the s-th node has t^s neighbors, the communication complexity of sharing dictionaries is $\mathcal{O}(t^s MKd)$, while the complexity of sharing ϵ and N^s can be omitted since they are numbers. Therefore, the overall communication complexity of each node is $\mathcal{O}(t^s MKd)$, which is independent to N^s.

5. Experiments

We evaluate the proposed DCQ on different datasets for approximate nearest neighbor (ANN) search and image retrieval. The training data is randomly distributed to different nodes in a network. We construct a network with 10 simulated nodes, as shown in Figure 1. Our machine is equipped with 24 Intel Xeon CPUs E5-2630 (2.30GHz) and 96 GB memory. We empirically set the penalty parameter $\rho = 100$ and the number of ADMM iterations $I = 5$. Note that $\rho = 100$ is less than the theoretical threshold in our proof that guaranntees convergence of ADMM (7.19×10^{10}), however, the theoretical threshold is sufficient but not necessary since it is an extremely loose bound. As shown in our supplementary material, $\rho = 100$ leads to better accuracy and faster convergence than the theoretical threshold, and we didn't observe non-convergence for $\rho = 100$.

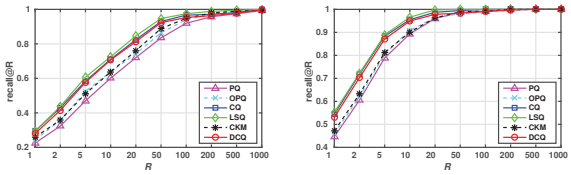
Ideally, the resulting local dictionaries $\{\mathrm{C}^s\}$ among all the nodes will be consistent because they are learned with the consensus constraints. However, since the solution obtained by ADMM is not theoretically guaranteed to be the global optimum (Boyd et al. 2011), the consistency of the local dictionaries needs to be verified. Therefore, we come up with two search strategies for the proposed DCQ. In the first strategy, each node will keep its local dictionary and composition codes after the optimization is done. Subsequently, the distances between the query and database vectors are calculated using the local lookup table. In the second strategy, we assign one of the local dictionaries as the final dictionary C, then update the composition codes $\{\mathrm{B}^s\}$ one more iteration to finally end the learning. We empirically find the results of these two variants are very close, which implies the consistency of the learned local dictionaries and the convergence of ADMM in our algorithm. In the following comparisons, we simply report the performance of the first strategy.

To our best knowledge, this work is the first attempt to learn quantization based algorithm in a distributed setting, therefore we cannot find closely related work to compare with in this setting. Thus, we compare our Distributed Composite Quantization against several state-of-the-art centralized methods, namely, Product Quantization (PQ) (Jegou, Douze, and Schmid 2011), Optimized Product Quantization (OPQ) (Ge et al. 2013), Cartesian k-means (CKM) (Norouzi and Fleet 2013), Composite Quantization (CQ) (Zhang, Du, and Wang 2014) and iterated Local Search for AQ



(a) MNIST dataset. (b) LabelMe22K dataset.

Figure 2: The recall@R for different algorithms on MNIST and LabelMe22K datasets with 64-bit codes.



(a) 64-bit codes. (b) 128-bit codes.

Figure 3: The recall@R for different algorithms on SIFT1M dataset with 64 and 128 bits.

(LSQ) (Martinez et al. 2016). All the algorithms perform linear scan search using asymmetric distance (Jegou, Douze, and Schmid 2011).

5.1. Evaluation on ANN Search

We perform the ANN search experiments on three datasets: MNIST (LeCun et al. 1998), 28×28 grayscale images of handwritten digits, which we treat as 784-dimensional vectors; LabelMe22K (Russell et al. 2008), a corpus of images expressed as 512-dimensional GIST descriptors; and SIFT1M (Jegou, Douze, and Schmid 2011), which consists of $1M$ 128-dimensional SIFT features as base vectors, $100K$ learning vectors and $10K$ queries. Following the convention (Zhang, Du, and Wang 2014; Martinez et al. 2016), we measure the search quality using recall@R, *i.e.*, for varying values of R, the average rate of queries for which the 1-nearest neighbor is ranked in the top R positions. The ground-truth nearest neighbors are computed over the original features using linear scan.

Figure 2 shows the comparison on MNIST and LabelMe22K dataset with 64-bit codes. On both datasets, our method achieves comparable or even higher accuracy compared with centralized baselines. It can be observed that the performance of our algorithm is very close to that of CQ, which suggests that learning composite quantizers in a distributed setting does not compromise much quality compared to the centralized version. Also, our method and CQ outperform the state-of-the-art partition-based quantization algorithms such as PQ and OPQ, which validates the advantages of addition-based quantizations.

Figure 3 shows the results on a larger dataset SIFT1M using 64-bit and 128-bit codes. These results are consistent with the findings in Figure 2. The curves of our DCQ and CQ almost overlap, which again validates the efficacy of our generalization of CQ to the distributed settings.

	#Bits	PQ	OPQ	CKM	LSQ	CQ	DCQ
Fisher	32	0.451	0.469	0.497	0.505	0.501	0.503
	64	0.471	0.492	0.538	0.564	0.560	0.554
	128	0.496	0.517	0.568	0.610	0.602	0.595
VLAD	32	0.484	0.493	0.506	0.512	0.508	0.506
	64	0.519	0.526	0.548	0.577	0.572	0.575
	128	0.538	0.562	0.576	0.619	0.614	0.610

Table 1: The mAP on the Holidays dataset with distractors using different code lengths.

	#Bits	PQ	OPQ	CKM	LSQ	CQ	DCQ
Fisher	32	1.457	1.586	1.899	2.076	2.043	2.017
	64	1.637	1.822	2.194	2.354	2.336	2.315
	128	1.883	2.031	2.334	2.450	2.434	2.406
VLAD	32	1.639	1.788	1.906	2.172	2.126	2.072
	64	1.925	2.030	2.203	2.421	2.357	2.341
	128	2.088	2.177	2.349	2.644	2.584	2.587

Table 2: The performance over the UKBench dataset in terms of scores using different length of codes encoding.

5.2. Evaluation on Image Retrieval Applications

We also evaluate our method on the application of compact codes (Perronnin et al. 2010) to image retrieval on the INRIA Holidays dataset (Jegou, Douze, and Schmid 2008) that contains 500 queries and 991 corresponding relevant images, and the UKBench dataset (Nister and Stewenius 2006) that contains $10,200$ images of $2,550$ groups with four images each. Following common practice (Jegou, Douze, and Schmid 2008; 2011), we use extra one million MIRFlickr-1M images (Huiskes and Lew 2008) as distracters.

We follow (Jegou, Douze, and Schmid 2008) to evaluate the performance over the INRIA Holidays dataset with mean average precision (mAP), *i.e.*, the mean area under the precision-recall curve; The evaluation metric on the UKBench dataset is provided in (Nister and Stewenius 2006), which is the score indicating how many of the other images are in the top-4 rank where one image of each group is used as query. We use 4096-dimensional Fisher vectors (Perronnin and Dance 2007) and VLAD vectors (Jégou et al. 2010) as the image descriptors for both datasets.

The search results are shown in Table 1 and 2. One can see that the performance of our approach DCQ is competitive and close to CQ, overall better than PQ, OPQ and CKM, but not as good as LSQ. However, all listed methods except our DCQ are centralized approaches and cannot work in distributed case. Also, addition-based approaches like LSQ and CQ still show superior accuracies to partition-based ones such as PQ and OPQ, which is consistent to what we have observed in ANN search experiments in Section 5.1.

5.3. Evaluation on Convergence and Efficiency

Convergence We empirically find that our algorithm is able to fast converge to a local minima. To show this, we measure the objective function value of Equation (5) at each iteration and compare it with the objective function value

of Composite Quantization, *i.e.*, Equation (4). We test DCQ and CQ (Zhang, Du, and Wang 2014) on the SIFT1M dataset with the same initialization using 64-bit codes. When the difference of current objective value and last objective value is less than 1% of current objective value, we consider them as convergent and stop training. The values of Equation (5) and (4) are comparable since both of them measure the reconstruction loss plus the constant inter-dictionary-element-product loss. If our distributed learning algorithm works perfectly such that all C across all nodes are equal in each iteration, then Equation (5) and (4) will have the same value.

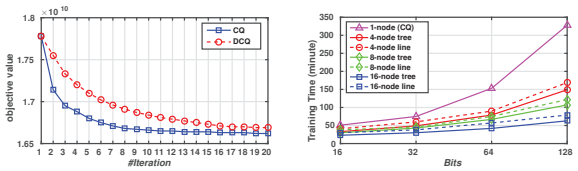
As shown in Figure 4(a), the objective value of DCQ at each iteration always decreases, and converges around 20 iterations. Moreover, the objective value of DCQ is not much higher than that of the original CQ, which demonstrates the power of the proposed distributed learning scheme.

Efficiency Although both DCQ and CQ require around 20 iterations to converge, our DCQ could largely reduce the training time by exploiting parallel computation. To quantitatively show the efficiency advantages of DCQ, we evaluate the training time of DCQ and CQ on the SIFT1M dataset using different code lengths. We also vary the number of nodes to see how training time can be shortened with more nodes. When the same number of nodes are employed, the training time might be affected by the topology of the network. For convenience, we choose two representative topologies in this experiments, *i.e.*, binary tree topology and line topology.

Due to the lack of a real computing cluster, we measure the training time by simulating each node sequentially on a single machine. To be more specific, each simulated node solely occupies the entire machine in turn for training on its local data, *e.g.*, updating the local dictionaries, and the time cost for the slowest node to update its local dictionaries will be regarded as the training time for the simulated cluster to update all local dictionaries. Similar protocols are also applied to obtain the time for updating B and ϵ. The communication time is neglected due to the small communication complexity as analyzed in Section 4.4.2.

Figure 4(b) presents the training time under different settings. Roughly speaking, the training time seems to increase quadratically as the code length increases. Meanwhile, the more nodes involved in computation, the shorter the training time. This phenomenon is intuitive. With more nodes involved in the computation, the size of data distributed to each node becomes smaller, thus the corresponding training time is less. Take 128-bit code length as an example, the training time of DCQ with 16-node binary tree topology is about 63 minutes on the learning set of SIFT1M of $100,000$ samples, while CQ takes 328 minutes using a single machine. Also, it could be observed that the line topology takes a few more minutes to converge than binary tree topology, but the total training time is still acceptable considering that line topology is already the most undesiable case among all topologies. These results demonstrate the potentials of the proposed DCQ for massive data in real-world applications.

Note that the number of iterations to convergence, the objective value at convergence, and the recall@R in testing phase are slightly different under different settings. We measure those 3 quantities on SIFT1M dataset using 64 bits code. As shown in Table 3, those 3 quantities do not vary a lot with the network topology or the number of nodes, which demonstrates the robustness of DCQ. For the number of iterations to convergence, line topology takes more iterations to converge than binary tree topology, probably due to the larger diameter of line topology. However, we didn't observe correlations between the objective value at convergence and the recall@R in testing phase, and the network topology or the number of nodes.



(a) Convergence curves (b) Training time

Figure 4: Convergence and training time on SIFT1M dataset. (a) The vertical axis stands for the objective function value, the horizontal axis for the number of iterations. (b) The vertical axis stands for the training time (minute), the horizontal axis for different code lengths.

No. of nodes / topology	1	4		8		16	
		tree	line	tree	line	tree	line
No. of iters to converge	19	22	26	20	24	21	26
objective value ($\times 10^{10}$)	1.66253	1.66259	1.66351	1.66303	1.66389	1.66358	1.66344
recall@1 (/%)	54.43	53.27	52.78	53.86	53.45	52.19	52.55

Table 3: The results are obtained on SIFT1M dataset using 128-bit codes.

6. Conclusion

In this paper, we propose Distributed Composite Quantization (DCQ) to learn quantizers and composition codes for distributed data. We cast the centralized Composite Quantization (Zhang, Du, and Wang 2014) into a set of decentralized subproblems with consensus constraints and showed how these subproblems can be solved in parallel in a distributed manner. Our method could adapt to arbitrary network topologies with low communication and computational cost. Extensive experiments on several large-scale datasets verify the efficacy of our method.

Acknowledgements

This work is supported in part by Singapore Ministry of Education Academic Research Fund Tier 2 MOE2015-T2-2-114 and was carried out at the Rapid-Rich Object Search (ROSE) Lab in the Nanyang Technological University, Singapore. The ROSE Lab is supported by the National Research Foundation, Singapore, under its Interactive Digital Media (IDM) Strategic Research Programme. We gratefully acknowledge the support of NVAITC (NVIDIA AI Technology Centre) for their donation of a Tesla K80 and M60 GPU used for our research at the ROSE Lab.

The authors appreciate Xueyan Tang, Ting Zhang and Jingdong Wang for their kind help!

References

Babenko, A., and Lempitsky, V. 2014. Additive quantization for extreme vector compression. In *CVPR*.

Babenko, A., and Lempitsky, V. 2015. Tree quantization for large-scale similarity search and classification. In *CVPR*.

Bertsekas, D. P. 1989. *Parallel and distributed computation: numerical methods*. Prentice hall Englewood Cliffs, NJ.

Bhattacharjee, S. D.; Yuan, J.; Hong, W.; and Ruan, X. 2016. Query adaptive instance search using object sketches. In *Proceedings of the 2016 ACM on Multimedia Conference*.

Boyd, S.; Parikh, N.; Chu, E.; Peleato, B.; and Eckstein, J. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*.

Cong, Y.; Yuan, J.; and Liu, J. 2011. Sparse reconstruction cost for abnormal event detection. In *CVPR*.

Corbett, J. C.; Dean, J.; Epstein, M.; Fikes, A.; Frost, C.; Furman, J. J.; Ghemawat, S.; Gubarev, A.; Heiser, C.; Hochschild, P.; et al. 2013. Spanner: Googles globally distributed database. *ACM Transactions on Computer Systems*.

Ge, T.; He, K.; Ke, Q.; and Sun, J. 2013. Optimized product quantization for approximate nearest neighbor search. In *CVPR*.

Heo, J.-P.; Lin, Z.; and Yoon, S.-E. 2014. Distance encoded product quantization. In *CVPR*.

Hestenes, M. R. 1969. Multiplier and gradient methods. *Journal of optimization theory and applications*.

Hong, M.; Luo, Z.-Q.; and Razaviyayn, M. 2016. Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems. *SIAM Journal on Optimization*.

Hong, W.; Meng, J.; and Yuan, J. 2018. Tensorized projection for high-dimensional binary embedding. In *AAAI*.

Hong, W.; Yuan, J.; and Das Bhattacharjee, S. 2017. Fried binary embedding for high-dimensional visual features. In *CVPR*.

Huiskes, M. J., and Lew, M. S. 2008. The mir flickr retrieval evaluation. In *Proceedings of the 1st ACM international conference on Multimedia information retrieval*. ACM.

Jégou, H.; Douze, M.; Schmid, C.; and Pérez, P. 2010. Aggregating local descriptors into a compact image representation. In *CVPR*.

Jegou, H.; Douze, M.; and Schmid, C. 2008. Hamming embedding and weak geometric consistency for large scale image search. In *ECCV*.

Jegou, H.; Douze, M.; and Schmid, C. 2011. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*.

Leng, C.; Wu, J.; Cheng, J.; Zhang, X.; and Lu, H. 2015. Hashing for distributed data. In *ICML*.

Liang, J.; Zhang, M.; Zeng, X.; and Yu, G. 2014. Distributed dictionary learning for sparse representation in sensor networks. *IEEE Transactions on Image Processing*.

Liu, D. C., and Nocedal, J. 1989. On the limited memory bfgs method for large scale optimization. *Mathematical programming*.

Liu, B.; Yuan, X.-T.; Yu, Y.; Liu, Q.; and Metaxas, D. N. 2016. Decentralized robust subspace clustering. In *AAAI*.

Liu, X.; Li, Z.; Deng, C.; and Tao, D. 2017. Distributed adaptive binary quantization for fast nearest neighbor search. *IEEE Transactions on Image Processing*.

Martinez, J.; Clement, J.; Hoos, H. H.; and Little, J. J. 2016. Revisiting additive quantization. In *ECCV*.

Meng, J.; Wang, H.; Yuan, J.; and Tan, Y.-P. 2016. From keyframes to key objects: Video summarization by representative object proposal selection. In *CVPR*.

Nister, D., and Stewenius, H. 2006. Scalable recognition with a vocabulary tree. In *CVPR*.

Nocedal, J. 1980. Updating quasi-newton matrices with limited storage. *Mathematics of computation*.

Norouzi, M., and Fleet, D. J. 2013. Cartesian k-means. In *CVPR*.

Perronnin, F., and Dance, C. 2007. Fisher kernels on visual vocabularies for image categorization. In *CVPR*.

Perronnin, F.; Liu, Y.; Sánchez, J.; and Poirier, H. 2010. Large-scale image retrieval with compressed fisher vectors. In *CVPR*.

Powell, M., and Authority, U. K. A. E. 1967. *"A method for non-linear constraints in minimization problems"*. Atomic Energy Res. Estab. Theoretical Physics Div. ; AERE TP 310. U.K.A.E.A.

Russell, B. C.; Torralba, A.; Murphy, K. P.; and Freeman, W. T. 2008. Labelme: a database and web-based tool for image annotation. *International journal of computer vision*.

Shakhnarovich, G.; Indyk, P.; and Darrell, T. 2006. *Nearest-neighbor methods in learning and vision: theory and practice*.

Wang, X.; Zhang, T.; Qi, G.-J.; Tang, J.; and Wang, J. 2016a. Supervised quantization for similarity search. In *CVPR*.

Wang, Z.; Duan, L.-Y.; Yuan, J.; Huang, T.; and Gao, W. 2016b. To project more or to quantize more: minimizing reconstruction bias for learning compact binary codes. In *IJCAI*.

Wang, J.; Zhang, T.; Sebe, N.; Shen, H. T.; et al. 2017. A survey on learning to hash. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Zhang, R., and Kwok, J. 2014. Asynchronous distributed admm for consensus optimization. In *ICML*.

Zhang, T., and Wang, J. 2016. Collaborative quantization for cross-modal similarity search. In *CVPR*.

Zhang, T.; Qi, G.-J.; Tang, J.; and Wang, J. 2015. Sparse composite quantization. In *CVPR*.

Zhang, T.; Du, C.; and Wang, J. 2014. Composite quantization for approximate nearest neighbor search. In *ICML*.