Product Quantizer Aware Inverted Index for Scalable Nearest Neighbor Search

Haechan Noh, Taeho Kim, Jae-Pil Heo*
Sungkyunkwan University
{noru0114, kth0522, jaepilheo}@skku.edu

Abstract

The inverted index is one of the most commonly used structures for non-exhaustive nearest neighbor search on large-scale datasets. It allows a significant factor of acceleration by a reduced number of distance computations with only a small fraction of the database. In particular, the inverted index enables the product quantization (PQ) to learn their codewords in the residual vector space. The quantization error of the PQ can be substantially improved in such combination since the residual vector space is much more quantization-friendly thanks to their compact distribution compared to the original data. In this paper, we first raise an unremarked but crucial question; why the inverted index and the product quantizer are optimized separately even though they are closely related? For instance, changes on the inverted index distort the whole residual vector space. To address the raised question, we suggest a joint optimization of the coarse and fine quantizers by substituting the original objective of the coarse quantizer to end-to-end quantization distortion. Moreover, our method is generic and applicable to different combinations of coarse and fine quantizers such as inverted multi-index and optimized PQ.

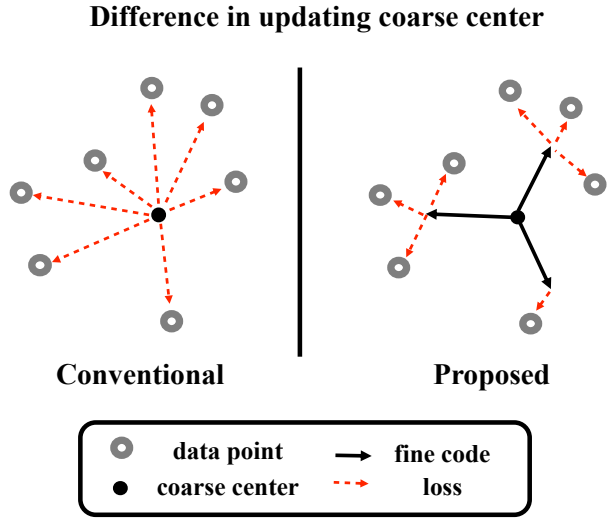


Figure 1: This figure describes difference between coarse center updates of conventional inverted index and those of our proposed method. While the conventional inverted index considers the distortion of the coarse centers only, the proposed method also considers the quantization error of the fine quantizer in the coarse center updates.

1. Introduction

For decades, approximate nearest neighbor search has been a vital problem in various fields including computer vision. It is especially difficult with high-dimensional and large-scale datasets because of its impractical requirements for computational costs and memory overhead. To address such difficulty, efficient indexing and compact data representation techniques have been highlighted.

The Product Quantization (PQ) [16] and its variations [11, 19, 3, 29, 15, 8, 23, 21] has been recognized as the most popular and successful solution, since they provide significant factors of data compression rate and efficiency in distance estimation. Specifically, the PQ divides the high-dimensional vector into M disjoint sub-vectors and quantizes those sub-vectors into K sub-codewords independently. In this way, the PQ utilizes M^K representations with a small memory footprint. Moreover, it is empirically shown that the PQ-based techniques provide superior search quality than parallel research directed toward the same goal, the binary hashing methods [2, 27, 14, 28, 13, 9]

The inverted index provides a non-exhaustive search by a simple shortlisting mechanism based on the data clustering. A practical scalable search system is implemented by the following procedure; given a query, the inverted index collects the shortlist which is a small fraction of the whole data, and the PQ re-ranks the candidates according to the estimated distances with the compact codes. Behind this visible synergy effect between them towards high scalability in terms of both speed and memory, the inverted index transforms the data into quantization-friendly residual vec-

*Corresponding author

tors suitable for the PQ. Since the residual vector has a much more compact distribution compared to its original data, the PQ benefits from it to further reduce the quantization distortion.

In this paper, we raise a new question about the aforementioned tight relationship between the coarse (inverted index) and fine (product) quantizers; why they are learned separately? For instance, a change in the coarse quantizer significantly affects the distribution of residual vectors. As a result, we suggest that coarse and fine quantizers should be jointly and collaboratively optimized. To this end, we propose a joint optimization of the coarse and fine quantizers by substituting the objective of the coarse quantizer to end-to-end quantization distortion. Finally, the designed scheme is orthogonal to the conventional inverted index techniques without any additional time and memory overhead at encoding, decoding, and query time. The experimental validation with various indexing and encoding techniques on several benchmarks are available in Sec. 5.

Our main contributions are summarized as follows:

- We highlight the necessity of joint optimization of the coarse and fine quantizers.
- We propose a joint optimization of coarse and fine quantizers by replacing the objective of coarse quantizer into the distortion of the fine quantizer.
- Our proposed method is orthogonal to the conventional inverted index techniques without any additional time overhead at encoding, decoding, and query time.
- Experimental results show that our method achieves state-of-the-art performances over the five large-scale ANN datasets.

2. Related Work

2.1. Product Quantization

Product quantization (PQ) [16] is a vector quantization method that aims to compress data by learned representative codewords. Usually, the PQ is used in large-scale datasets that require extortionate memory size. The PQ divides an input vector into several sub-vectors and applies vector quantization on each sub-vectors. Also, the PQ allows efficient computation of Euclidean distances between the uncompressed query and the large number of compressed vectors, which is called Asymmetric Distance Computation (ADC) via an instance lookup table.

The original PQ divides an input vector uniformly without considering the correlation among dimensions. This method works well with structured features like SIFT, but it is not always the best way. To manage the input vector efficiently, the original PQ paper proposed to apply a random rotation for all vectors preliminary to make dimensions uncorrelated. Besides, the author of [17] proposed to optimize an orthogonal matrix by a set of reflection with a variance balancing criterion. The Optimized Product Quantization (OPQ) [11] is an extension of these ideas. The OPQ computes a rotation matrix that minimizes the quantization error iteratively by formulating an Orthogonal Procrustes Problem [12].

While the original PQ represents the vector as a concatenation of the codewords from sub-dimensions, Additive Quantization (AQ) [3] and Composite Quantization (CQ) [29] represent a vector as the sum of codewords whose dimensionality is identical to the original vector. These addition-based methods are a generalization of the PQ, and the PQ is a special case of the AQ where the codewords from different codebooks are orthogonal.

2.2. Inverted Index System with PQ

Although the linear scan with Asymmetric Distance Computation (ADC) of the PQ improved search speed, the search is still exhaustive. To handle over the very large-scale database, [16] proposed a search system with inverted indexing [25], which is called IVFADC. It groups the database into several disjoint subsets by K-means clustering and allows accelerated search by comparing distance with only a small fraction of the database. For each coarse group residual vector, the displacement between the given vector and the center of a group it belongs to, is encoded by the PQ.

Locally Optimized Product Quantization (LOPQ) [19] improves the performance of the IVFADC by locally defining the product quantizer for each subset of the database independently. However, it requires much larger memory consumption and slows down the retrieval because of its multiple quantizers.

The Inverted Multi Index (IMI) [4] is a multi-dimensional extension of IVFADC. IMI decomposes data space into a Cartesian product of two sub-spaces and clusters each sub-space independently. Then the subsets of the database are defined by the pair of indices from each sub-space. Representing subsets by the combination of indices enables a much finer partition of search space without increasing query time.

The idea of IVFADC encoding residual vectors is similar to Residual Vector Quantization (RVQ) [7] where residual from the previous quantization step is further encoded recursively. Its improved versions [6, 1] optimized fine codewords across different steps jointly. However, in both IVFADC and RVQ, the optimization of the coarse and fine quantizers remains independent whereas they are highly related. Moreover, the joint optimization techniques are confined to exhaustive retrieval task, and not verified on non-exhaustive retrieval task which is crucial to deal with large-scale datasets.

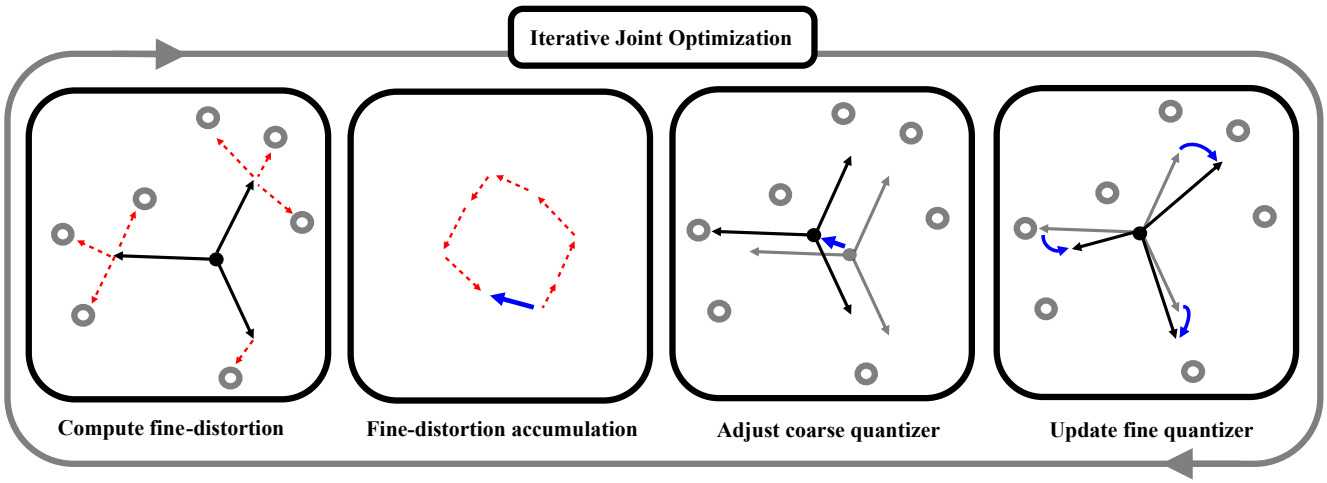


Figure 2: Illustration of proposed joint optimization of the coarse and fine quantizer. The grey circles represent data points. The black solid arrow and red dashed arrow indicates fine codeword and fine-distortion, respectively. The blue arrow shows the direction of the updating. The overall procedure of the optimization consists of four steps. First, compute the fine-distortion with proposed objective (Eq. 9). Then, accumulate the computed fine-distortion (Eq. 11). Next, adjust the coarse quantizer with the accumulated error vector (Eq. 10). Last, update the fine quantizer with adjusted coarse indices. Consequently, the coarse and fine quantizers are optimized collaboratively.

3. Background and Motivations

3.1. Background

Let us briefly review the Product Quantization (PQ) and the IVFADC, and define notations that we will use throughout the paper. When D-dimensional database $X = \{x_n\}_{n=1}^{N}, x_n \in \mathbb{R}^D$ and a query $y \in \mathbb{R}^D$ is given, finding nearest neighbor of y is formulated as follow:

$$n = \underset{n \in \{1,\dots,N\}}{\arg\min} ||y - x_n||^2. \tag{1}$$

Eq. 1 takes $O(DN)$ of time complexity and $4DN$ bytes of memory consumption. To manage this time and memory usage, the PQ proposed a compact code encoding method to trade-off retrieval accuracy against resources. First, given D-dimensional vector $x \in \mathbb{R}^D$ is divided into M sub-vectors, $x = [x^1, ..., x^M], x^i \in \mathbb{R}^{D/M}$. Next, product quantizer q_p is trained by clustering the sub-vectors of each subspace $m \in \{1, ..., M\}$ into set of K number of codeword $C^m = \{c_k^m\}_{k=1}^{K}$. Then, a vector x_n is encoded into PQ code, $q_p(x, n) = [i^1(x_n^1), ..., i^M(x_n^M)]$, as follows by the trained product quantizer q_p:

$$i^m(x_n^m) = \underset{k \in \{1,\dots,K\}}{\arg\min} ||x_n^m - c_k^m||^2. \tag{2}$$

The quantization distortion which represents the amount of lost information of the encoded vector can be measured as follows:

$$E(x_n, C) = \sum_{m=1}^{M} \min_{k \in \{1,\dots,K\}} ||x_n^m - c_k^m||^2. \tag{3}$$

While the product quantizer reduces the retrieval time significantly, it is highly suggested to combine the PQ and the inverted file system to avoid exhaustive search on the large-scale databases. Coarse quantizer q_c with K' number of coarse centers is learned by K-means clustering. Then residual vector $r(x) = x - q_c(x)$ between a vector from the database x and its nearest coarse center is utilized to training the product quantizer q_p which we refer to the fine quantizer. Finally, $\tilde{x}$ is a reconstruction of x and defined as follows:

$$\tilde{x} = q_p(x - q_c(x)) + q_c(x). \tag{4}$$

Approximated distance $d(\cdot, \cdot)$ between the database vector x and the given query y is computed as follows:

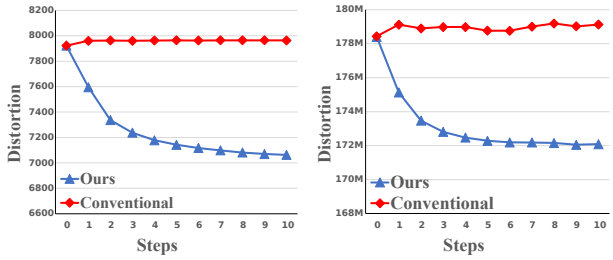
$$d(x, y) \approx d(\tilde{x}, y) = d(\tilde{x} - q_c(y), y - q_c(y)), \tag{5}$$

where the x and y are in same coarse cluster and $q_c(x) = q_c(y)$.

3.2. Motivation

Prior inverted index techniques utilize the K-means to cluster the inverted indices. The objective function of the K-means clustering is finding a partition of data S that minimize the following objective:

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x \in S_i} ||x - \mu_i||^2 \tag{6}$$



(a) SIFT-1M (b) Conv-1M

Figure 3: This figure shows the end-to-end quantization error of the conventional method and ours. Starting from a pre-trained inverted index system, q_c is updated by 10 iterations further and q_p retrained with updated q_c for each step. While joint optimization of coarse and fine quantizers improves the end-to-end distortion, the conventional objective remains around the initial error.

where μ_i is mean vector of the set S_i. Training the K-means clusters consists of the following two alternating steps:

$$S_i^{(t)} = \{x_p : ||x_p - \mu_i^{(t)}||^2 \leq ||x_p - \mu_j^{(t)}||^2 \, \forall_j, \, 1 \leq j \leq K'\}. \tag{7}$$

Assignment step to finding closest cluster for each vector, and center update step to minimize Eq. 6.

$$\mu_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j \tag{8}$$

Inverted indices or coarse codeword $\mu = [\mu_1, ..., \mu_{K'}]$ learned by the K-means algorithm play a crucial role in training the fine quantizer because the residual vector is defined by its nearest coarse codeword. However, this procedure of the K-means algorithm has no guarantee of minimizing the quantization error of the fine quantizer. We insist that the objective function of the coarse quantizer should consider not only the error of itself but also the error of the fine quantizer.

4. Our Approach

4.1. Coarse Objective to Minimize Fine-Error

To optimize coarse quantizer with considering end-to-end quantization distortion, we substitute the original objective function of the coarse quantizer into the objective function of the fine quantizer.

$$\arg\min_{S} \sum_{i=1}^{K'} \sum_{x \in S_i} ||q_p(x - \mu_i) - (x - \mu_i)||^2 \tag{9}$$

It minimizes the quantization error of the residual vector $r(x) = x - q_c(x)$ whereas the K-means algorithm only minimizes error between a vector x and its corresponding coarse center $q_c(x)$. Fig. 1 describes the difference between a conventional K-means objective function (Eq. 6) and the proposed one (Eq. 9).

Algorithm 1 Joint optimization process of coarse and fine quantizers

Input: coarse quantizer q_c, fine quantizer q_p, number of coarse centers K', scaling factor s, learning set X, number of maximum optimization steps T.
Output: Optimized q_c and q_p.

procedure JOINT OPTIMIZATION
  for i=1,...,T **do**
    with fixed fine quantizer:
      UpdateCoarseQuantizer()
    with fixed coarse quantizer:
      UpdateFineQuantizer()

procedure UPDATE COARSE QUANTIZER
  Compute initial quantization error (Eq. 3).
  while True **do**
    for i=1,...,K' **do**
      Compute mean error vector (Eq. 11).
      Update coarse centers (Eq. 10).
    Re-assignment step (Eq. 7).
    Compute quantization error (Eq. 3).
    if Quantization error is converged. **then**
      break

procedure UPDATE FINE QUANTIZER
  Compute residual vectors (Eq. 12).
  Training Fine quantizer with the residual vectors.

Similar to the K-means, the proposed objective (Eq. 9) is trained by two alternating steps: assignment and update. We modified the update step for the coarse indices $C' = [c'_1, ..., c'_{K'}]$ as follows:

$$c_i'^{(t+1)} = c_i'^{(t)} + s * E_i, \tag{10}$$

where s is a scaling factor, and E_i is the mean error vector of the i-th coarse center.

$$E_i = \frac{1}{|S_i|} \sum_{x \in S_i} \{(x - q_c(x)) - q_p(x - q_c(x))\} \tag{11}$$

We start the initial state from trained K-means clustering.

After the update step (Eq. 10), vectors from the learning set X is re-assigned with the updated coarse quantizer in the assignment step (Eq. 7). These procedures are repeated until the proposed objective function is converged.

Method	K'	64bit			128bit		
		R@1	R@10	R@100	R@1	R@10	R@100
IVFADC	2^{10}	0.2962	0.7036	0.9566	0.4714	0.9026	**0.9930**
Ours	2^{10}	**0.3108**	**0.7301**	**0.9652**	**0.4964**	**0.9277**	0.9921
IVFOADC	2^{10}	0.2952	0.7194	0.9626	0.4752	0.9125	**0.9945**
IVFOADC + Ours	2^{10}	**0.3214**	**0.7555**	**0.9715**	**0.5001**	**0.9311**	0.9930
Multi-D-ADC	2^{8}	0.3027	0.7291	0.9673	0.4843	0.9242	**0.9980**
Multi-D-ADC + Ours	2^{8}	**0.3181**	**0.7482**	**0.9766**	**0.5129**	**0.9338**	0.9973

Table 1: Comparison on SIFT-1M dataset.

Method	K'	64bit			128bit		
		R@1	R@10	R@100	R@1	R@10	R@100
IVFADC	2^{10}	0.1998	0.5814	0.8959	0.3214	0.7580	0.9585
Ours	2^{10}	**0.2028**	**0.5977**	**0.9083**	**0.3318**	**0.7848**	**0.9729**
IVFOADC	2^{10}	0.1978	0.5828	0.9109	0.3228	0.7783	0.9706
IVFOADC + Ours	2^{10}	**0.2072**	**0.6094**	**0.9215**	**0.3386**	**0.8037**	**0.9788**
Multi-D-ADC	2^{8}	0.1833	0.5686	0.9032	0.3090	0.7604	0.9702
Multi-D-ADC + Ours	2^{8}	**0.1943**	**0.5822**	**0.9143**	**0.3219**	**0.7905**	**0.9780**

Table 2: Comparison on Conv-1M dataset.

4.2. Joint Optimization of Coarse and Fine Qauntizer.

In the conventional inverted index system, the coarse and fine quantizers are trained independently. First, the coarse quantizer is trained with the K-means clustering algorithm. Then, the fine quantizer is trained using the residual between the coarse index and the database vector.

We introduce the joint optimization of the coarse and fine quantizers. Our optimization scheme is illustrated in Fig. 2. Our objective function (Eq. 9) updates the coarse quantizer to minimize the quantization error of the fine quantizer. Then, the residual vector between the coarse index and the database vector are redefined as follow:

$$r(x)^{(t+1)} = x - q_c^{(t+1)}. \tag{12}$$

Thus, we can further minimize the quantization error by retraining the fine quantizer to be tailored with the redefined residual vectors. Consequently, the coarse quantizer also can be optimized further to minimize distortion of the redefined fine quantizer. This sequence of optimizing is repeated iteratively. The overall procedure of our method is summarized in Algorithm 1.

We conduct an experiment to measure quantization distortion by adding ten extra iterations for the coarse quantizer and the redefining fine quantizer as shown in Fig. 3. Starting from the same pre-trained inverted index system, the quantization error with our objective is reduced significantly. However, the distortion of the conventional objective remains around the initial level regardless of additional clustering iterations. This experiment justifies that our collaborative optimization improves the traditional method.

4.3. Extension to Inverted Multi Index

The Inverted Multi Index (IMI) introduced a new inverted index structure that enables much fine indexing without increasing query time. It decomposes vector $X \in \mathbb{R}^{\mathbb{D}}$ into two halves, where $X^1 \in \mathbb{R}^{\frac{D}{2}}$ and $X^2 \in \mathbb{R}^{\frac{D}{2}}$. Then it computes the inverted index on X^1 and X^2 independently, and produces two coarse codebooks $U = [u_1, ...u_{K'}]$ and $V = [v_1, ..., v_{K'}]$. The database is divided into $K' \times K'$ number of disjoint subsets while the number of comparisons between the coarse centers and query vector remains $K' + K'$.

The proposed method can be applied to the IMI orthogonally. To this end, the update step can be modified as follows:

$$\begin{aligned} u_i^{(t+1)} &= u_i^{(t)} + s * E_i^u, \\ v_i^{(t+1)} &= v_i^{(t)} + s * E_i^v \end{aligned} \tag{13}$$

where E^u and E^v is mean error vector from $X^1, X^2 \in \mathbb{R}^{\frac{D}{2}}$, respectively.

Method	K'	64bit			128bit		
		R@1	R@10	R@100	R@1	R@10	R@100
IVFADC	2^{10}	0.0780	0.2190	0.4850	0.1350	0.3410	0.6670
Ours	2^{10}	**0.0900**	**0.2410**	**0.4990**	**0.1470**	**0.3670**	**0.6830**
IVFOADC	2^{10}	0.1380	0.3660	0.7010	0.1860	0.5420	**0.8790**
IVFOADC + Ours	2^{10}	**0.1470**	**0.3780**	**0.7290**	**0.2100**	**0.5610**	0.8770
Multi-D-ADC	2^{8}	0.1250	0.3640	**0.7390**	0.2340	0.5630	0.8860
Multi-D-ADC + Ours	2^{8}	**0.1430**	**0.3880**	0.7370	**0.2390**	**0.5820**	**0.8960**

Table 3: Comparison on GIST-1M dataset.

Method	K'	R@1	R@10	R@100
IVFADC	2^{13}	0.0973	0.3410	0.6744
Ours	2^{13}	**0.1143**	**0.3848**	**0.7088**
Multi-D-ADC	2^{10}	0.1253	0.3732	0.6786
Multi-D-ADC + Ours	2^{10}	**0.1372**	**0.3769**	**0.6878**
Multi-D-ADC	2^{11}	0.1317	0.3658	0.6360
Multi-D-ADC + Ours	2^{11}	**0.1449**	**0.3790**	**0.6381**

Table 4: Comparison on SIFT-1B dataset

Method	K'	R@1	R@10	R@100
IVFADC	2^{13}	0.1738	0.3886	0.6361
Ours	2^{13}	**0.1751**	**0.3925**	**0.6527**
Multi-D-ADC	2^{10}	0.1645	**0.3538**	0.5887
Multi-D-ADC + Ours	2^{10}	**0.1666**	0.3529	**0.5907**
Multi-D-ADC	2^{11}	0.1706	0.3603	0.5908
Multi-D-ADC + Ours	2^{11}	**0.1734**	**0.3612**	**0.5931**

Table 5: Comparison on Deep-1B dataset

5. Evaluation

5.1. Protocol

We evaluate our method on following benchmarks:

- **SIFT-1M [16]**: It contains one million 128-dimensional SIFT local descriptor [20] vectors for database, 100K learning set, and 10K query vectors.
- **SIFT-1B [16]**: It is extended version of the SIFT-1M dataset. It contains one billion vectors for database, 0.1 billion learning set. We use 500K vectors randomly sampled from the learning set for training.
- **Deep-1B [5]**: It contains 96-dimensional DNN features from fully connected layer of GoogLeNet [26] architecture for a billion images on the Web.
- **Conv-1M**: We randomly sample 1M images for database, 100K for learning set and 10K for query from ImageNet database [10]. Then, we extract 512-dimensional global average pooled feature from conv5 layer of the VGG network [24].
- **GIST-1M [16]**: 960-dimensional global color GIST descriptors [22], it has one million database, 500K learning set and 1K query vectors. We randomly selected 100K vectors for training from the learning set.

We compare the following combinations of coarse and fine quantizers:

- **IVFADC**: Inverted file system [25] based on the Product Quantization [16].
- **IVFOADC**: IVFADC with the OPQ [11].
- **Ours**: Proposed joint optimization of the coarse and fine quantization with the IVFADC.
- **IVFOADC + Ours**: Proposed joint optimization of the coarse and fine quantization with the IVFOADC [11].
- **Multi-D-ADC**: Inverted Multi Index [4] with the PQ.
- **Multi-D-ADC + Ours**: Proposed joint optimization of the coarse and fine quantization with the Multi-D-ADC.

The retrieval performance is measured by average recall@R, the proportion of queries that have their first nearest neighbor of uncompressed space within top-R nearest neighbors in quantized space.

We fixed hyper-parameters for all experiments except ablation study and billion scale datasets, $M = \{8, 16\}$ and $K = 256$ for product quantizer, the number of optimization steps $T = 10$ and scaling factor $s = 0.1$. The number of coarse indices is set to $K' = 2^{10}$ and $K' = 2^{8}$ for IVFADC and IMI, respectively on SIFT-1M, Conv-1M, and GIST-1M. Note that the IMI has K' indices for each subspace, and the database is divided into $K' \times K'$ number of disjoint subsets. For each query, we guarantee the number of candidates L at least 50k.

Method	L	R@1	R@10	R@100
IVFADC	0.1M	0.0809	0.2560	0.4248
Ours		**0.0933**	**0.2730**	**0.4330**
IVFADC	0.5M	0.0954	0.3255	0.6160
Ours		**0.1110**	**0.3622**	**0.6340**
IVFADC	5M	0.0985	0.3490	0.7147
Ours		**0.1153**	**0.3996**	**0.7685**
IVFADC	10M	0.0985	0.3493	0.7166
Ours		**0.1155**	**0.4002**	**0.7714**
Multi-D-ADC	1K	0.0770	0.1831	0.2446
Multi-D-ADC+Ours		**0.0791**	**0.1936**	**0.2600**
Multi-D-ADC	10K	0.0979	0.2608	0.3604
Multi-D-ADC+Ours		**0.1009**	**0.2735**	**0.3856**
Multi-D-ADC	0.1M	0.1202	0.3700	0.6115
Multi-D-ADC+Ours		**0.1298**	**0.3715**	**0.6260**
Multi-D-ADC	0.5M	0.1250	0.3753	0.6774
Multi-D-ADC+Ours		**0.1348**	**0.3789**	**0.6849**
Multi-D-ADC	5M	0.1262	0.3689	0.6594
Multi-D-ADC+Ours		**0.1393**	**0.3744**	**0.6700**

Table 6: Ablation study for L on SIFT-1B 64bits encoding

Method	K'	R@1	R@10	R@100
IVFADC	2^9	0.2838	0.6973	0.9517
Ours		**0.2985**	**0.7229**	**0.9574**
IVFOADC		0.2918	0.7098	0.9587
IVFOADC + Ours		**0.3170**	**0.7355**	**0.9683**
IVFADC	2^{10}	0.2962	0.7036	0.9566
Ours		**0.3108**	**0.7301**	**0.9652**
IVFOADC		0.2952	0.7194	0.9626
IVFOADC + Ours		**0.3214**	**0.7555**	**0.9715**
IVFADC	2^{11}	0.2902	0.7021	0.9574
Ours		**0.3086**	**0.7370**	**0.9684**
IVFOADC		0.3090	0.7236	0.9657
IVFOADC + Ours		**0.3290**	**0.7548**	**0.9774**
Multi-D-ADC	2^7	0.3027	0.7268	0.9677
Multi-D-ADC + Ours		**0.3182**	**0.7453**	**0.9768**
Multi-D-ADC	2^8	0.3027	0.7291	0.9673
Multi-D-ADC + Ours		**0.3181**	**0.7482**	**0.9766**
Multi-D-ADC	2^9	0.3166	0.7427	0.9723
Multi-D-ADC + Ours		**0.3280**	**0.7703**	**0.9795**
Multi-D-ADC	2^{10}	0.3397	0.7636	0.9799
Multi-D-ADC + Ours		**0.3446**	**0.7850**	**0.9845**

Table 7: Ablation study for K' on SIFT-1M 64bits encoding

5.2. Results on million scale datasets

Table. 1, Table. 2, and Table. 3 report recall@R scores of non-exhaustive ANN search on the SIFT-1M, Conv-1M, and GIST-1M, respectively. Generally, the method with **Ours** shows better results than without it. For example, **Ours**, **IVFOADC + Ours**, and **Multi-D-ADC + Ours** on 64bits encoding improve its conventional counterpart for 4.93%, 8.88%, and 5.09% in terms of R@1 as shown in Table. 1. Moreover, the performance improvement of R@1 scores is larger than R@10 and R@100. On SIFT-1M and Conv-1M dataset, **Ours** is even superior to not only **IVFADC**, but also **IVFOADC** without combining with OPQ. As shown in Table. 2, IVFOADC improves IVFADC only 2.24% while the proposed method increases its conventional counterpart 3.77% and 5.46 with and without OPQ respectively in terms of R@10 score on 64bits encoding. The experimental results verify that the proposed method consistently improves with various combinations of coarse and fine quantizers, thus it is orthogonal to conventional inverted index systems.

5.3. Results on billion scale datasets

For billion scale datasets, we set $K' = 2^{13}$ and $K' = \{2^{10}, 2^{11}\}$ for IVFADC and IMI, respectively. The number of minimum candidates L is set to one million for the billion scale database. Table. 4 and Table. 5 report the performance comparison on SIFT-1B and Deep-1B. The results show that the proposed method is applicable to the billion scale dataset and the improvement is even superior than the million scale datasets. For instance, the improvement of R@1 score of **Ours** over **IVFADC** is 17.47% on SIFT-1B whereas it is 4.93% on SIFT-1M. Moreover, the performance gains of **Multi-D-ADC+Ours** are 9.49% and 10.02% on $K' = 2^{10}$ and $K' = 2^{11}$, respectively. They are about twice the gain on SIFT-1M.

5.4. Ablation study

We conduct ablation studies with respect to two parameters: the number of coarse indices K' and the minimum number of candidates L. First, we verifies our method on $L = \{0.01\%, 0.05\%, 0.1\%, 0.5\%, 1\%\}$ while fixing other parameters. For example, 1M number of candidates are 0.1% of a billion-scale database. The result of ablation study for L on SIFT-1B dataset is summarized in Table. 6. Regardless of the size of L, the proposed method consistently improves the performance. Especially, the performance gap increases with large L in terms of R@10 and R@100. For instance, proposed method improves the R@100 score about 7.65% with $L = 10M$ while it is 1.93% and 2.92% with $L = 0.1M$ and $L = 0.5M$ respectively. Second, an ablation study for the number of coarse

Method	K'	Fixed L, avg recall	Fixed L, avg W	Fixed W, avg recall	Fixed W, avg L
IVFADC	2^8	0.9840	12.6501	0.9848	54242.7205
Ours		0.9805	12.2584	0.9786	56314.9056
IVFADC	2^{10}	0.9965	48.8316	0.9963	52370.7237
Ours		0.9942	48.5948	0.9933	52592.1584
IVFADC	2^{12}	0.9992	181.7226	0.9990	56188.6639
Ours		0.9983	187.8200	0.9984	53996.8614

Table 8: Pruning quality study on SIFT-1M 64bits encoding.

indices is conducted on the SIFT-1M dataset. We verifies our method on $K' = \{2^9, 2^{10}, 2^{11}\}$ on IVFADC and IVFOADC based method, and $K' = \{2^7, 2^8, 2^9, 2^{10}\}$ on IMI based method while fixing other parameters as $M = 8$ and $L' = 50,000$. As reported in Table. 7, the proposed method consistently improves the performance regardless of the number of coarse indices K'.

5.5. Indexing Quality Analysis

Although the proposed method has identical time complexity with the IVFADC, an imbalance of the inverted lists can slow down the actual retrieval time. For instance, if the coarse clusters have uneven cluster size, some queries should recompute their residual vector and lookup table many times to guarantee the sufficient number of candidates. To validate proposed method does not harm the quality or balance of the inverted lists, we conduct indexing quality analysis. We measure an average number of accessed inverted-lists W with fixed minimum candidate size $L = 50,000$, and an average number of candidates L with a fixed number of retrieved inverted-lists W. Recall with fixed L or W indicates whether the ground truth is included in the candidate list regardless of its rank. Table. 8 shows the experimental results and it verifies that the proposed method does not harm the indexing quality of the coarse index. The differences of average recall between **IVFADC** and **Ours** are much less than 1% for both fixed L and fixed W. The average number of accessed inverted-lists W and average number of candidates L differ less than 5%.

5.6. Overhead Analysis

Note that the proposed method does not have any additional overhead in encoding, decoding, and search time, but training time. With learning set X where $|X| = N$, the training time complexity of the proposed method is described by:

$$O(T * u * ND(K + K')) \tag{14}$$

where T is a number of optimization steps, u is an average number of iterations of learning the coarse quantizer. Time and memory consumption of the encoding, decoding, and retrieval is identical to the IVFADC.

Method	K	L	search time (ms)
IVFADC	2^{13}	1M	38.9356
Ours			39.2435
IVFADC	2^{13}	5M	178.7390
Ours			176.1460
Multi-D-ADC	2^{10}	1M	44.4053
Multi-D-ADC + Ours			49.8225
Multi-D-ADC	2^{10}	5M	165.1767
Multi-D-ADC + Ours			160.8871

Table 9: Average search times on SIFT-1B dataset.

Table. 9 demonstrates the practical average retrieval time per query on SIFT-1B dataset with FAISS framwork [18]. The **IVFADC** and **Ours** shows negligible search time difference.

6. Conclusion

The problem of the conventional inverted index system that its coarse and fine quantizers optimized in a non-collaborative way is first pointed out in this paper. To address the problem, we have proposed a joint optimization of the coarse and fine quantizers by replacing the original objective function of the inverted index into the distortion of the fine quantizer. The proposed method can be orthogonally applied to the conventional inverted index techniques including the IVFADC, IVFOADC, and IMI without any time and memory overhead on encoding, decoding, and searching time. We have evaluated our method with various combinations of the coarse and fine quantizers on several benchmarks, and the advantage of the proposed method is consistently demonstrated over the baselines.

Acknowledgments This work was supported in part by MCST/KOCCA (No. R2020070002), MSIT/IITP (No. 2020-0-00973, 2019-0-00421, 2020-0-01821, and 2020-0-01550), MSIT/NRF (No. NRF-2020R1F1A1076602), and MSIT&KNPA/KIPoT (Police Lab 2.0, No. 210121M06).

References

[1] Liefu Ai, Junqing Yu, Zebin Wu, Yunfeng He, and Tao Guan. Optimized residual vector quantization for efficient approximate nearest neighbor search. *Multimedia Systems*, 23(2):169–181, 2017.

[2] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *2006 47th annual IEEE symposium on foundations of computer science (FOCS'06)*, pages 459–468, 2006.

[3] Artem Babenko and Victor Lempitsky. Additive quantization for extreme vector compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 931–938, 2014.

[4] Artem Babenko and Victor Lempitsky. The inverted multi-index. *IEEE transactions on pattern analysis and machine intelligence*, 37(6):1247–1260, 2014.

[5] Artem Babenko and Victor Lempitsky. Efficient indexing of billion-scale datasets of deep descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2055–2063, 2016.

[6] W-Y Chan, Smita Gupta, and Allen Gersho. Enhanced multistage vector quantization by joint codebook design. *IEEE Transactions on Communications*, 40(11):1693–1697, 1992.

[7] Yongjian Chen, Tao Guan, and Cheng Wang. Approximate nearest neighbor search by residual vector quantization. *Sensors*, 10(12):11259–11273, 2010.

[8] Xinyan Dai, Xiao Yan, Kelvin KW Ng, Jiu Liu, and James Cheng. Norm-explicit quantization: Improving vector quantization for maximum inner product search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[9] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262, 2004.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255, 2009.

[11] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. Optimized product quantization for approximate nearest neighbor search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2946–2953, 2013.

[12] 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, 2012.

[13] Kaiming He, Fang Wen, and Jian Sun. K-means hashing: An affinity-preserving quantization method for learning binary compact codes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2938–2945, 2013.

[14] Jae-Pil Heo, Youngwoon Lee, Junfeng He, Shih-Fu Chang, and Sung-Eui Yoon. Spherical hashing. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[15] Jae-Pil Heo, Zhe Lin, and Sung-Eui Yoon. Distance encoded product quantization. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[16] 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, 2010.

[17] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 3304–3311. IEEE, 2010.

[18] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 2019.

[19] Yannis Kalantidis and Yannis Avrithis. Locally optimized product quantization for approximate nearest neighbor search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2321–2328, 2014.

[20] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[21] Julieta Martinez, Joris Clement, Holger H Hoos, and James J Little. Revisiting additive quantization. In *European Conference on Computer Vision*, pages 137–153. Springer, 2016.

[22] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International journal of computer vision*, 42(3):145–175, 2001.

[23] Ezgi Can Ozan, Serkan Kiranyaz, and Moncef Gabbouj. Competitive quantization for approximate nearest neighbor search. *IEEE Transactions on Knowledge and Data Engineering*, 28(11):2884–2894, 2016.

[24] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[25] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *null*, page 1470. IEEE, 2003.

[26] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[27] Antonio Torralba, Rob Fergus, and Yair Weiss. Small codes and large image databases for recognition. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[28] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In *Advances in neural information processing systems*, pages 1753–1760, 2009.

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