A STUDY ON PARAMETER TUNING FOR OPTIMAL INDEXING ON LARGE SCALE DATASETS

Dinh-Nghiep Le[*], **Van-Thi Hoang**[†], **Duc-Toan Nguyen**[‡], **The-Anh Pham**[§]
[*] Hong Duc University (HDU)
[†]Department of Education and Training, Thanh Hoa city
[‡] Department of Industry and Trade, Thanh Hoa city
[§] Hong Duc University (HDU)

Abstract—**Fast matching is a crucial task in many computer vision applications due to its computationally intensive overhead, especially for high feature spaces. Promising techniques to address this problem have been investigated in the literature such as product quantization, hierarchical clustering decomposition, etc. In these approaches, a distance metric must be learned to support the re-ranking step that helps filter out the best candidates. Nonetheless, computing the distances is a much intensively computational task and is often done during the online search phase. As a result, this process degrades the search performance. In this work, we conduct a study on parameter tuning to make efficient the computation of distances. Different searching strategies are also investigated to justify the impact of coding quality on search performance. Experiments have been conducted in a standard product quantization framework and showed interesting results in terms of both coding quality and search efficiency.**

Index Terms—**Feature indexing, Approximate nearest neighbor search, Product quantization**

I. Introduction

With the increasing development of social networks and platforms, the amount of data in multimedia applications grows rapidly in both scale and dimensional aspects. Indexing and searching on these billion-scale and high-dimensional datasets become a critical need as they are fundamental tasks of any computer vision system. In this field, objects are mostly unstructured and usually unlabeled. It is thus very hard to compare them directly. Instead, the objects are represented by real-valued, high-dimensional vectors and some distance metrics must be employed to perform the feature matching. In most situations, it is impractical for multimedia applications to perform exact nearest neighbor (ENN) search because of expensively computational cost. Therefore, fast approximate nearest neighbor (ANN) search is much preferred in practice to quickly produce (approximate) answers for a given query with very high accuracy ($> 80\%$).

As the key techniques for addressing the ANN search problem, product quantization (PQ) [1] and its optimized variations [2], [3], [4] have been well studied and demonstrated promising results for large-scale datasets. In its essence, the PQ algorithm first decomposes the high-dimensional space into a Cartesian product of low dimensional sub-spaces and then quantizes each of them separately. Since the dimensionality of each subspace is relatively small, using a small-sized codebook is sufficient to obtain the satisfied searching performance. Although computational cost can be effectively reduced, the PQ method is subjected to the key assumption that the sub-spaces are mutually independent. To deal with this problem, several remedies have been proposed to optimize the quantization stage by minimizing coding distortion such as Optimized Product Quantization (OPQ) [3], ck-means, [4], local OPQ [5]. In the former methods, OPQ and ck-means, the data is adaptively aligned to characterize the intrinsic variances. Codebook learning is jointly performed with data transformation to achieve the independence and balance between the sub-spaces. As a result, quantization error is greatly reduced, yielding better fitting to the underlying data. Nonetheless, these methods are still less effective for the case of multi-model distribution feature spaces. Latter method, like local OPQ, aims at handling this issue by first decomposing the data into compact and single-model groups, followed by applying the OPQ process for each local cluster. Alternatively, other tree quantization methods, e.g., Additive Quantization (AQ) [6], Tree Quantization (TQ) [7], have been presented to deal with the mutual independence assumption of PQ. Differing from the PQ spirit, these methods does not divide the feature space into smaller sub-spaces. They instead encode each input vector as the sum of M codewords coming from M codebooks. Moreover, the codewords in AQ and TQ are of the same length as the input vectors, but many components are set to zero in each codeword of TQ. As the sub-space independence assumption is omitted, the AQ and TQ-based methods give better coding accuracy than PQ but they are not superior to PQ in terms of search speedups [7].

Recently, hierarchical clustering decomposition methods [8], [9], [10] have been extensively utilized as an embedding fashion with the PQ framework. In the hierarchical clustering approach [11], a clustering algorithm is iteratively applied to partition the feature vectors into smaller groups. The entire decomposition can be well represented by a tree structure that works as an inverted file structure for driving the search process. Different attempts [12], [13]

Correspondence: The-Anh Pham
email: phamtheanh@hdu.edu.vn
Manuscript received: 6/2020, revised: 9/2020, accepted: 10/2020.

have incorporated the clustering tree with a priority queue, resulting in an effective search strategy. Combining the benefits of hierarchical clustering idea and product quantization, the work in [8] has proposed an unified scheme and substantially improved the ANN search performance. Later improvements [9], [10] focus on optimizing the coding quality by introducing the concept of semantic sub-space decomposition. As such, the data space is divided into sub-spaces or sub-groups, each of which contains elements closing to each other. Product quantization is then performed for each sub-group. The resulting quantization quality has been significantly improved.

One of the main difficulties posed in a product quantization scheme is concerned with the use of a distance metric to construct a short-list of candidate answers for a given query. In the literature, two kinds of distance metrics are often employed, symmetric distance computation (SDC) and asymmetric distance computation (ADC). The former approximates the distance between two points by the (Euclidean) distance between their quantization codewords. In contrast, the latter measures the distance of two points as how far a point is from the quantization codeword of the other point. From the definition, it is obvious to observe that the ADC gives a better approximation of the Euclidean distance than the SDC does. However, this favored property comes at a computational cost. The ADC distances must be computed during the online searching phase, while the SDC is not. In fact, the SDC metric can be pre-computed using the lookup tables when the codebook is learned. In this work, we favor the use of SDC measurements to improve the search timings, while still expecting a high level of coding quality. To meet this double-goal question, we propose first to employ the hierarchical product quantization (HPQ) scheme [9] to achieve the minimal construction error. We then perform different studies to derive the best parameter tuning for effective usage of the SDC distance. To validate the propositions, extensive experiments have been conducted and showed interesting results.

For the remainder of this paper, Section 2 reviews the main points of PQ method, HPQ as well as hierarchical vocabulary clustering tree. Section 3 describes the experiment protocol, datasets, and evaluation results. Finally, Section 5 draws some key remarks and discusses the follow-up works.

II. System architecture

In this work, it is denoted that X is a dataset in the D-dimensional feature vector space (R^D) and for a given vector $x \in R^D$, let $a_j(x)$ with $1 \le j \le m$ be the operator that returns a sub-vector of x starting from the j^{th} dimension to $(j+h)^{th}$ dimension where $h = D/m - 1$, here m is an integer such that D is a multiple of m. Given a vector $x \in R^D$, one can employ $a_j(x)$ to split x into m disjoint sub-vectors $\{a_1(x), a_2(x), \ldots, a_m(x)\}$, each of which has the length of D/m.

In the PQ method [1], a learning dataset X is divided into m disjoint sub-spaces in the way as the operator $a_j(x)$ does. For each sub-space, a clustering algorithm is then applied to learn a codebook composing of K codewords or clusters (typically, $m = 8$ and $K = 256$). Each codeword has length of D/m. Given an input vector $x \in R^D$, the quantization of x is done by dividing x into m sub-vectors followed by finding the nearest codeword of each sub-vector in the corresponding codebook. Specifically, a quantization operator $q_j(x)$ is defined in the j^{th} sub-space as follows:

$$q_j(x) \leftarrow \underset{1 \le k \le K}{\arg\min}\, \mathbf{d}(a_j(x), c_{j,k}) \tag{1}$$

where $c_{j,k}$ is the k^{th} codeword of the codebook constructed from the j^{th} sub-space, and $\mathbf{d}$ is the Euclidean distance function.

With the $q_j(x)$ defined above, quantization of x is a m-dimensional integer vector formed by concatenating the quantization in each sub-space:

$$q(x) \leftarrow \{q_1(x), q_2(x), \ldots, q_m(x)\}. \tag{2}$$

For convenience of presentation, we also denote that:

$$\hat{q}_j(x) \leftarrow \underset{c_{j,k}}{\arg\min}\, \mathbf{d}(a_j(x), c_{j,k}) \tag{3}$$

with $1 \le k \le K$. That means $\hat{q}_j(x)$ outputs the codeword closest to the sub-vector $a_j(x)$ in the j^{th} sub-space.

PQ uses both SDC and ADC distances for re-ranking the candidates. Mathematically, the SDC distance between two points $x, y \in R^D$ is formulated as follows:

$$\mathbf{d}_{SD}(x,y) = \sum_{j=1}^{m} \mathbf{d}(\hat{q}_j(x), \hat{q}_j(y)), \tag{4}$$

while, the ADC distance is approximately computed by:

$$\mathbf{d}_{AD}(x,y) = \sum_{j=1}^{m} \mathbf{d}(a_j(x), \hat{q}_j(y)). \tag{5}$$

It is worth noting in the PQ scheme that the sub-spaces are grouped with the same order as in the original space. Hence, it is probably not ensured that the resulting sub-spaces are mutually independent and balanced (in terms of variance). These criteria are needed for yielding good coding quality. Furthermore, the codebooks in different sub-spaces may contain similar codewords due to the similarity in visual content which appears at different positions in a scene. It thus does not meet the assumption of mutual independence and also raises the question of redundancy in bit allocation for the codewords.

To address these issues, we have recently proposed a novel coding quantization scheme known as hierarchical product quantization (HPQ) [9]. In contrast to PQ, space decomposition is done in such a way that similar data points shall enter into one sub-space. As such, the points in each sub-space are highly correlated, while the two different sub-spaces are mutually independent. In particularly, HPQ algorithm can be sketched as follows:

- Divide the database $X \in R^D$ into m sub-spaces ($m = 8$) as the PQ does.
- Apply a clustering algorithm for the data in *all* the sub-spaces to form m sub-groups.
- Train a codebook (each has K codewords) for the data contained in each sub-group.

When the codebooks are learned, quantizing a vector $x \in R^D$ is proceeded in two steps: finding the closest sub-group for each sub-vector of x and finding the closest codeword in the corresponding sub-group. Algorithm 1 outlines the main steps of this process. As the sub-groups are constructed by a clustering process, it is obvious to see that they are mutually independent and distinctive (i.e., the data in each sub-group are highly correlated). Due to its natural process, we consider each sub-group as a semantic sub-space for codebook learning. This nice property helps yield high coding quality. However, when applied to ANN search task, the query time is impacted by the two-step quantization process as described above. Furthermore, HPQ is also subjected to the expensive cost of distance computation, especially for the ADC distance.

Algorithm 1 HPQuantizer(x, S, C)

```
1:  Input: An input vector ($x \in \mathbb{R}^D$), list of $m$ sub-groups ($S$) each has a center $S_j$, and the list of $m$ codebooks ($C$) each has $K$ codewords.
2:  Output: The quantization code of $x$ (i.e., $\mathbf{q}(x)$).
3:  $m \leftarrow length(S)$ {the number of sub-groups}
4:  split $x$ into $m$ sub-vectors: $a_1(x), a_2(x), \ldots, a_m(x)$
5:  $c_j \leftarrow 0$ for $j = 1, 2, \ldots, m$ {Initiated values of HPQ code}
6:  for each $a_j(x)$ do
7:      $h \leftarrow \arg\min_{1 \le i \le m} \mathbf{d}(a_j(x), S_i)$ {find the closest sub-group}
8:      $c_j^* \leftarrow \arg\min_{1 \le i \le K} \mathbf{d}(a_j(x), C_h(i))$ {$C_h(i)$: the $i^{th}$ codeword of the $h^{th}$ codebook}
9:      $c_j \leftarrow h \times K + c_j^*$
10: end for
11: return $\{c_j\}$
```

In the present work, we investigate an extension of HPQ and study the impact of different parameters to the coding quality. In the favor of SDC distance, we aim at deriving the best usage of pre-computed lookup tables so that the system can produce excellent ANN search performance.

Finer space decomposition: To use effectively the SDC metric, it is needed to give more effort for optimizing the coding quality of the codebooks. One can employ a strong method for this task such as ck-means [4], OPQ [3] but it comes at the cost of heavily computational overhead and thus can degrade the search timings. In our study, we propose to divide the feature space into finer sub-spaces for alleviating the impact of curse-of-dimensionality (i.e., $m = 16$ the number of sub-spaces). On the other hand, it is not necessary to use a high number of codewords for each codebook. By default, the number of codewords is set to $K = 256$ in most of the works in the literature [4], [3], [9], [1]. In the current study, we investigate the impact of coding quality by varying the parameter K in the collection of $\{32, 64, 128, 192\}$. By using lower codewords, it gives the computational benefit for both online and offline phases. The analytical computation cost of the quantization step (i.e., Algorithm 1) is characterized as: $m \times (m+K)$. That is the number of times the Euclidean distance operation $\mathbf{d}()$ is invoked.

It is worth noting that the dimensionality of the sub-space is also attributed to the complexity of quantization process. For instance, in the PQ method ($m = 8$), the Euclidean distance function $\mathbf{d}()$ operates in R^{16} and R^{120} sub-spaces for 128D SIFT and 960D GIST feature sets[1], respectively. When setting parameter $m = 16$, HPQ divides the feature space into finer sub-spaces resulting in less computation of the distance function. For a summary, Table I gives a picture of quantization complexity between PQ method and Algorithm 1 for several values of K accompanying the dimensionality of sub-spaces for SIFT and GIST features. One can observe that by varying the parameters m and K, HPQ does not incur much computation cost compared to the standard PQ method. In terms of coding quality, we shall provide detailed justification in the experimental section.

Bảng I
THE NUMBER OF TIMES CALLING THE DISTANCE OPERATOR $\mathbf{d}()$ FOR THE QUANTIZATION PROCESS

Method	SIFT	GIST	$K = 64$	$K = 128$	$K = 256$
PQ ($m = 8$)	R^{16}	R^{120}	-	-	2048
HPQ ($m = 16$)	R^{8}	R^{60}	1280	2304	4352

Efficient quantization with partial distance search: To further alleviate the computational overhead of the quantization process (e.g., our two-step quantization), we incorporate the use of partial distance search (PDS) [14] that helps terminate early the process of finding the closest codewords. In its essence, PDS performs unrolling the loop of distance computation in high dimensional space. By comparing the current (partial) distance value with the best distance established so far, it can decide to terminate early the loop. Algorithm 2 embeds the PDS idea into the computation of distance operator.

Algorithm 2 $D_{pds}(x, y, d_{best})$

```
1:  Input: Two input real vectors ($x, y$) and the best distance so far ($d_{best}$).
2:  Output: The (partial) distance between $x$ and $y$
3:  $n \leftarrow length(x)$ {$x$ and $y$ are the same dimensionlity}
4:  $d \leftarrow 0$
5:  for $j = 1, 2, \ldots, n$ do
6:      $a \leftarrow x(j) - y(j)$
7:      $d \leftarrow d + a \times a$
8:      if $d > d_{best}$ then
9:          return $d$ {terminate early if $d$ is not better than $d_{best}$}
10:     end if
11: end for
12: return $d$
```

With the PDS distance defined above, one can substitute the step of finding the closest center (i.e., lines 7 and 8 in Algorithm 1) by a more efficient procedure as follows (Algorithm 3):

[1]http://corpus-texmex.irisa.fr/

Algorithm 3 PDSQuantizer(x, L)

```
1: Input: An input vector ($x \in \mathbb{R}^n$) and a list $L$ containing centers or codewords in the sub-space $\mathbb{R}^n$.
2: Output: The center in $L$ closest to $x$.
3: $s \leftarrow length(L)$ {the size of the list $L$}
4: $i_{best} \leftarrow 1$ {Initiated value for the closest center}
5: $d_{best} \leftarrow \mathbf{d}(x, L(i_{best}))$ {Euclidean distance}
6: for $i = 2, \ldots, s$ do
7:     $d \leftarrow D_{pds}(x, L(i), d_{best})$ {PDS distance}
8:     if $d < d_{best}$ then
9:         $d_{best} \leftarrow d$
10:        $i_{best} \leftarrow i$
11:    end if
12: end for
13: return $i_{best}$
```

Incorporation of indexing clustering tree: Apart from improving the coding quality of the codebooks, it is needed to use an efficient indexing scheme to deal with the ANN search task. Hierarchical vocabulary clustering has been well studied in the past and achieved strong results when embedding into the product quantization fashion [8], [10]. In this study, we also employ this framework to perform ANN search. The search is optimized to obtain the highest speedup for a specific search precision. This was accomplished by a binary search procedure [15] which performs sampling on two parameters: the number of leaf nodes to visit and the size of the candidate short-list. In addition, as we use a higher value of m (i.e., $m = 16$ for obtaining finer space decomposition), it makes sense to apply the idea of PDS when compute the SDC distance between the query and the quantized samples in the database. As shall be shown in the experiments, this slight trick produces noticeable search speedups.

III. EXPERIMENTAL RESULTS

A. Datasets and evaluation metrics

In this section, we carry out a number of comparative experiments to validate the performance of our system in terms of both coding quality and search timings. For this purpose, state-of-the-art methods for coding and ANN search have been included in our study. These methods include FLANN library[2] [13], EPQ [8], Optimized EPQ (OEPQ) [16], HPQ [9], PQ [1] and the ck-means (i.e., Optimized PQ) [4]. For the evaluation datasets, we have chosen two benchmark feature sets: ANN_SIFT1M and ANN_GIST1M [1]. Detailed information of these datasets are given in Table II.

Bảng II
THE DATASETS USED FOR ALL THE EXPERIMENTS

Dataset	#Training	#Database	#Queries	#Dimension
ANN_SIFT1M	100,000	1,000,000	10,000	128
ANN_GIST1M	500,000	1,000,000	1000	960

As for the evaluation metrics, we employed the score Recall@R to measure the coding quality of the our system, PQ, and ck-means. These methods have been designed to minimize quantization errors. Here, Recall@R measures the fraction of corrected answers from a short-list of R candidates (typically $R = 1, 100, 1000$). For PQ and ck-means, we compute Recall@R for both SDC and ADC distances, whereas our system will be evaluated by using the SDC only. The goal here is to explore the marginal improvement of using finer sub-spaces. In addition, we also employed an other metric for measuring the search timings. Specifically, this matter can be well justified by using the search speedups/precisions curves as done in the literature [13], [17]. The speedups are relatively computed to sequence scan to avoid the impact of computer configuration. Search speedups are computed for a method $\mathcal{A}$ ($S_{\mathcal{A}}$) as follows:

$$S_{\mathcal{A}} = \frac{t_{Seq}}{t_{\mathcal{A}}} \tag{6}$$

where $t_{\mathcal{A}}$, t_{Seq} are the needed time to accomplish a given query of the method $\mathcal{A}$ and the brute-force search, respectively. For the stability, the search speedups and precisions are averaged for k queries, where $k = 10,000$ for SIFT and $k = 1000$ for GIST datasets. All the tests are run on a standard computer with following configuration: Windows 7, 16Gb RAM, Intel Core (Dual-Core) i7 2.1 GHz.

B. Results and discussions

This section is dedicated to the evaluation of all the studied methods for justifying the quality of codebooks and ANN search efficiency as well. We shall first present the results hereafter in terms of coding quality for the method: PQ, ck-means, and our HPQ method with varying parameters K (i.e., the number of codewords). For a summary, we report the parameter settings used in our tests as follows (Table III):

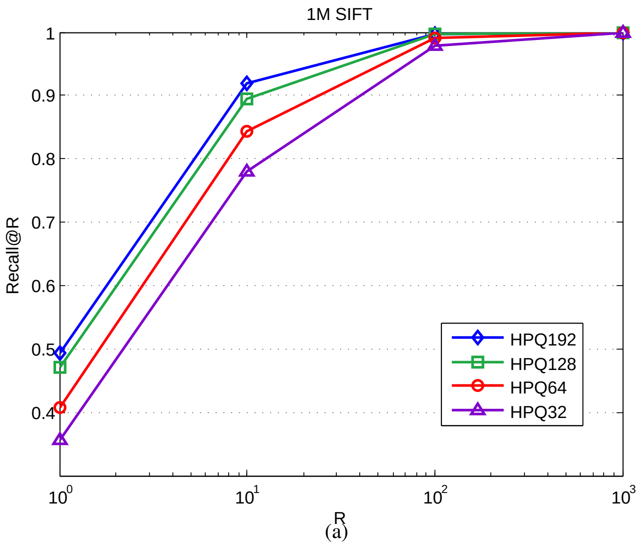
Bảng III
PARAMETERS USED IN OUR TESTS

Method	#sub-spaces (m)	#codewords (K)
PQ	8	256
ck-means	8	256
HPQ	16	$\{32, 64, 128, 192\}$

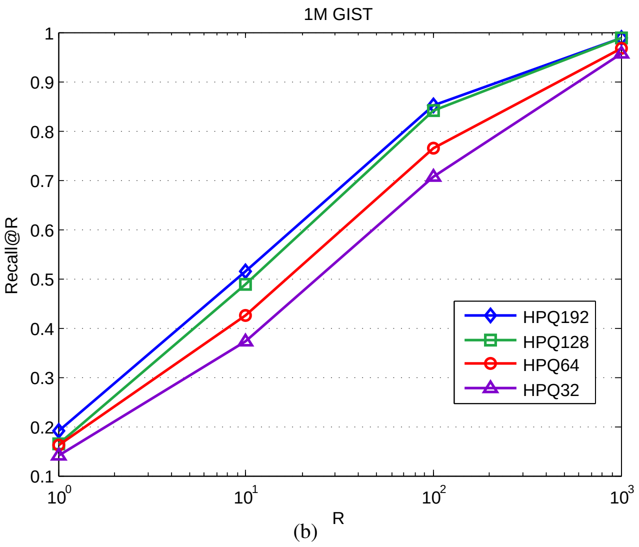
Figure 1 shows the Recall@R of our method with different settings of parameter K for both SIFT and GIST features using the SDC distance. As can be seen in the plots, coding recalls get increasing with respect to the high value of K. We have chosen the highest value of $K = 192$ so as to make it still lower than the default value used in PQ and ck-means ($K = 256$). In addition, one can also observe that the recall curves, corresponding to $K = \{128, 192\}$, operates on a par with each other for both feature datasets. This fact gives useful insights for the situations where one wishes to obtain the highest search speedups while expecting noticeable coding quality.

To have deeper insights of the proposed method, Figure 2 presents the comparative results with PQ and ck-means. In this evaluation, we selected the HPQ with $K = 32$ (the lowest performance curve, namely HPQ32) to be compared

[2]http://www.cs.ubc.ca/research/flann/

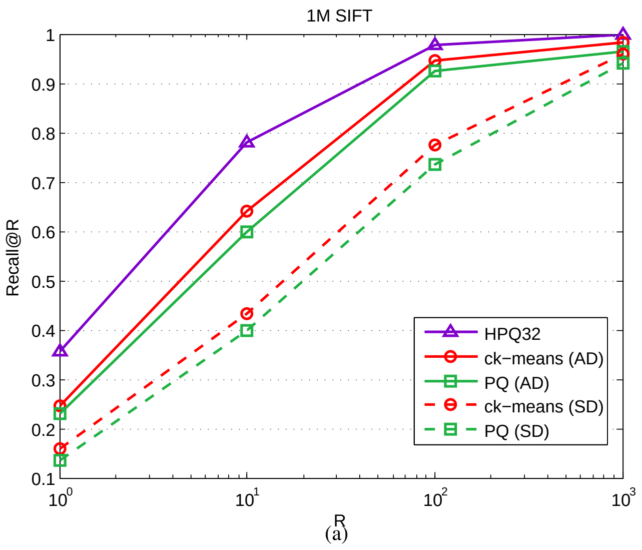


(a)

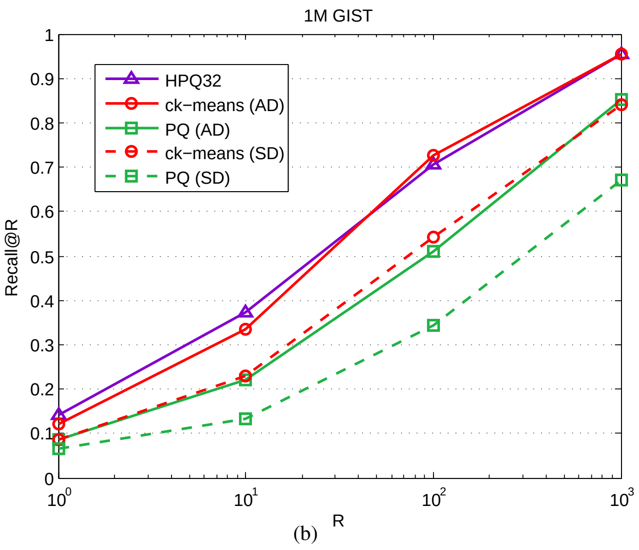


(b)

Hình 1. Coding quality of our system (HPQ) with varying number of codewords: (a) SIFT and (b) GIST features.



(a)

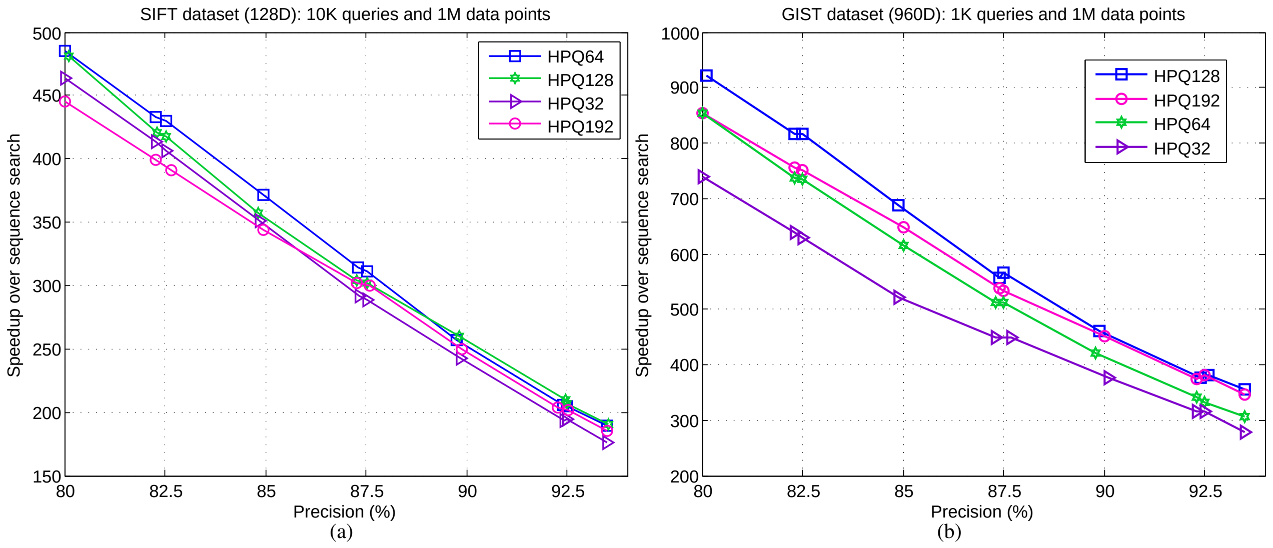


(b)

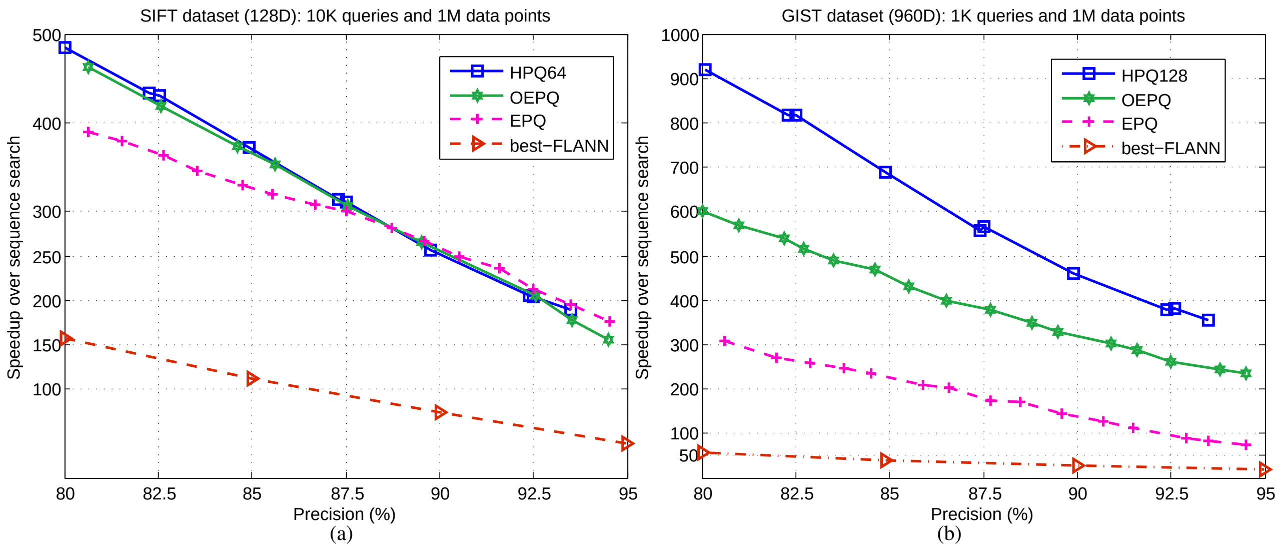
Hình 2. Coding quality of our system (HPQ32) and other methods: (a) SIFT and (b) GIST features.

with other methods. For the SIFT dataset, HPQ32 significantly outperforms all other methods for both ADC and SDC distances. It is worth mentioning that ck-means is a strong optimization version of the PQ method in the means of quantization quality but its performance (even for ADC distance) is much lower than that of HPQ32. This fact is very impressive when considering that HPQ32 uses a small number of codewords (i.e., 32 codewords for each codebook). When working on higher dimensional space (i.e., 960D GIST features), HPQ32 performs on a par with ck-means (ADC distance) version and is substantially superior to other methods. Connecting this outstanding performance of HPQ32 with the superior versions of HPQ presented previously (Figure 1), it can be concluded that by using finer sub-space decomposition, one can achieve significant benefit in terms of coding quality even the number of codewords is not many.

The results presented in Figures 1, 2 consistently confirm the expected quality of our method for the codebook learning. The remaining open question would be concerned with the search efficiency when applying to the ANN search task. In the following discussions, we shall continue to show the performance of our method for this task. Figure 3 presents the operating points of search speedups as a function of precision for all the HPQ versions in our study. For the SIFT dataset, the gap in performance is not that much for all the HPQ versions. In details, HPQ64 performs best in this case although its behavior is slightly superior to that of HPQ128. This observation is not fully synchronized for GIST dataset as shown in Figure 3(b). First, the performance gap is more noticeable, say for instances at $920\times$ and $732\times$ in speedups of HPQ128 and HPQ32, respectively, at the precision of 80%. Second, HPQ192 tends to be close to the winner (HPQ128), especially when considering very high search precisions ($> 90\%$). These new findings can be explained by the high dimensional space of GIST features in which coding quality plays a role to the success of search efficiency. As already noted in the Figure 1 (b), HPQ128 is virtually identical to HPQ192 in terms of



Hình 3. ANN search performance of system (HPQ) with varying number of codewords: (a) SIFT and (b) GIST features.



Hình 4. ANN search performance of our system and other methods: (a) SIFT and (b) GIST features.

coding quality, whereas HPQ128 incurs less computational overhead than HPQ192 does. As a result, HPQ128 gives the best search speedups in the studied experiments.

The last experiment has been conducted as shown in Figure 4 in which comparative search efficiency is provided for the best HPQ version (i.e., HPQ64 for SIFT and HPQ128 for GIST features), Optimized EPQ (OEPQ), EPQ, and the best method of FLANN (i.e., best-FLANN). It is worth highlighting that OEPQ is the state-of-the-art method for ANN search on the SIFT and GIST datasets [9], [16]. In this study, one can realize that HPQ64 can also reach the same level of search efficiency as OEPQ does for the SIFT dataset. Noticeably, using HPQ with 128 codewords provides substantial improvements for the GIST features. For instances, it gives a speedup of 921$\times$ compared to sequence scan when fixing the search precision of 80%. All these results confirm the superiority of our method, in terms of both coding quality and ANN search efficiency, especially when working in high dimensional spaces.

IV. CONCLUSIONS

In this work, a deep analysis and study of hierarchical product quantization has been conducted to examine its performance on the aspects of quantization quality and ANN search efficiency. Our proposal has been targeted to the fact that using finer space decomposition is essential for accomplishing these double-goal objective. Throughout extensive experiments in comparison with other methods, it was showed that our method provides significant improvement for various datasets and even tends to performs well with respect to the increase in space dimensionality. An interesting remark derived from our study is that a decent product quantizer can be constructed even without using a high number of codewords. As shown in our experiments, by using just as few as 32 codewords, one can also obtain satisfactory performance. Despite the obtained results are promising, we plan to investigate the inclusion of ADC distance as well as other deep learning based encoders for optimizing the method in follow-up works.

TÀI LIỆU THAM KHẢO

[1] H. Jegou, M. Douze, and C. Schmid, "Product Quantization for Nearest Neighbor Search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 1, pp. 117–128, 2011.
[2] A. Babenko and V. Lempitsky, "The inverted multi-index," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 6, pp. 1247–1260, 2015.
[3] T. Ge, K. He, Q. Ke, and J. Sun, "Optimized product quantization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 4, pp. 744–755, 2014.
[4] M. Norouzi and D. J. Fleet, "Cartesian k-means," in *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, ser. CVPR '13, 2013, pp. 3017–3024.
[5] Y. Kalantidis and Y. Avrithis, "Locally optimized product quantization for approximate nearest neighbor search," in *Proceedings of International Conference on Computer Vision and Pattern Recognition (CVPR 2014)*, Columbus, Ohio, June 2014, pp. 2329–2336.
[6] A. Babenko and V. Lempitsky, "Additive quantization for extreme vector compression," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 931–938.
[7] ——, "Tree quantization for large-scale similarity search and classification," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 4240–4248.
[8] T.-A. Pham and N.-T. Do, "Embedding hierarchical clustering in product quantization for feature indexing," *Multimedia Tools and Applications*, vol. 78, no. 1, pp. 9991–10 012, 2018.
[9] V.-H. Le, T.-A. Pham, and D.-N. Le, "Hierarchical product quantization for effective feature indexing," in *IEEE 26th International Conference on Telecommunications (ICT 2019)*, 2019, pp. 385–389.
[10] T.-A. Pham, D.-N. Le, and T.-L.-P. Nguyen, "Product sub-vector quantization for feature indexing," *Journal of Computer Science and Cybernetics*, vol. 35, no. 1, pp. 1–15, 2018.
[11] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, ser. CVPR'06, 2006, pp. 2161–2168.
[12] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *Proceedings of International Conference on Computer Vision Theory and Applications*, ser. VISAPP'09, 2009, pp. 331–340.
[13] ——, "Scalable nearest neighbor algorithms for high dimensional data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, pp. 2227–2240, 2014.
[14] J. McNames, "A fast nearest-neighbor algorithm based on a principal axis search tree," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 9, pp. 964–976, 2001.
[15] T.-A. Pham, S. Barrat, M. Delalandre, and J.-Y. Ramel, "An efficient tree structure for indexing feature vectors," *Pattern Recognition Letters*, vol. 55, no. 1, pp. 42–50, 2015.
[16] T.-A. Pham, "Improved embedding product quantization," *Machine Vision and Applications*, vol. 30, no. 3, pp. 447–459, 2019.
[17] ——, "Pair-wisely optimized clustering tree for feature indexing," *Computer Vision and Image Understanding*, vol. 154, no. 1, pp. 35–47, 2017.

NGHIÊN CỨU SỰ ẢNH HƯỞNG CỦA CÁC THAM SỐ TRONG TỐI ƯU HÓA CHỈ MỤC CHO CÁC CƠ SỞ DỮ LIỆU LỚN

Tóm tắt: Đối sánh nhanh là một trong những bài toán quan trọng của các ứng dụng thị giác máy bởi độ phức tạp tính toán lớn, đặc biệt là trong các không gian đặc trưng có số chiều lớn. Các kỹ thuật tiềm năng cho bài toán này đã được nghiên cứu và đề xuất trước đây như tích lượng tử (Product Quantization), thuật toán phân cụm phân cấp (Hierarchical Clustering Decomposition). Đối với các kỹ thuật này, một hàm khoảng cách sẽ được đề xuất để tạo một danh sách các ứng viên tiềm năng gần nhất với đối tượng truy vấn. Tuy nhiên, quá trình tính toán hàm khoảng cách này thường có độ phức tạp tính toán lớn và được thực hiện trong giai đoạn tìm kiếm (online), do vậy, làm ảnh hưởng đến hiệu năng tìm kiếm. Trong bài báo này, chúng tôi thực hiện các nghiên cứu trên các tham số ảnh hưởng đến quá trình lập chỉ mục và tối ưu hóa quá trình tính toán hàm khoảng cách. Ngoài ra, các chiến lược tìm kiếm khác nhau cũng được thực hiện để đánh giá chất lượng của quá trình lượng tử hóa. Các thử nghiệm đã được thực hiện và cho thấy những kết quả nổi bật cả về chất lượng lượng tử hóa và hiệu năng tìm kiếm.

Từ khóa: Lập chỉ mục, tìm kiếm xấp xỉ nhanh, tích lượng tử.

Dinh-Nghiep Le has been work at Hong Duc University as lecturer and permanent researcher since 2012. His research interests include: feature extraction and indexing, image detection and recognition, computer vision.

Van-Thi Hoang received his PhD thesis in 2006 from Hanoi National University of Education (Vietnam). He has been a lecturer at Hong Duc University until 2017. He has since then working at Department of Education and Training, Thanh Hoa city.

Duc-Toan Nguyen received the Master degree from University of Wollongong, Australia, in 2014. He has worked for Department of Industry and Trade since 2014, Thanh Hoa province. His research interests include: data mining, computer vision and machine learning.

The-Anh Pham has been working at Hong Duc University as a permanent researcher since 2004. He received his PhD Thesis in 2013 from Francois Rabelais university in France. Starting from June 2014 to November 2015, he has worked as a full research fellow position at Polytech's Tours, France. His research interests include document image analysis, image compression, feature extraction and indexing, shape analysis and representation.