Neurocomputing 362 (2019) 60–71

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Unsupervised deep quantization for object instance search

Wei Jiang[a], Yuwei Wu[a,*], Chenchen Jing[a], Tan Yu[b], Yunde Jia[a]

[a] Beijing Laboratory of Intelligent Information Technology, Beijing Institute of Technology, Beijing 100081, PR China
[b] Nanyang Technological University, Singapore

ARTICLE INFO

Article history:
Received 29 March 2019
Revised 14 June 2019
Accepted 20 June 2019
Available online 19 July 2019

Communicated by Yongdong Zhang

Keywords:
Image retrieval
Object instance search
Unsupervised deep quantization

ABSTRACT

In this paper, we propose an unsupervised deep quantization (UDQ) method for object instance search. The UDQ utilizes product quantization to discover the underlying self-supervision information of the training data and iteratively exploits the self-supervision information to optimize features of the training data in an unsupervised fashion. The optimized features are further used to update the self-supervision information for the subsequent training procedure. We introduce two constraints, the separability constraint and the discriminability constraint, to encourage the features to satisfy a cluster structure which is essential for the effective supervision information generation with the product quantization. The UDQ is optimized with an iterative optimization strategy which guarantees that the features and the supervision information can be enhanced each other alternately in a unified model. Moreover, we develop three refinement strategies to refine features to obtain better supervision information for the model optimization. Experimental results on four datasets show the superiority of our UDQ over the state-of-the-art methods.

1. Introduction

Nowadays, due to the explosive growth of image data, many advances have been made for image retrieval tasks [1–3]. In practice, an image often contains multiple objects [4–6], and each one may occupy a small portion of the image. Thus, traditional image retrieval methods measuring the whole image similarity with the query would fail to capture the relevance between the query and the small object. Motivated by users' demand for instance-level image retrieval, recently, object instance search has received increasing interests [7–9]. Given a query object, object instance search aims to retrieve all relevant images containing the specific object and localize the object in the retrieved images.

Since the diverse search requirements of different users are unpredictable and the expensive manual annotations are unaffordable, it is unrealistic to provide sufficient well-labeled training data to train an instance search model, especially for the large-scale scenario. Therefore, the object instance search is essentially an unsupervised task. This means that the applicability of supervised or semi-supervised approaches is rather limited for object instance search.

Recently, many unsupervised methods have been proposed for object instance search. Among them, most methods [10–12] utilize hand-crafted local features such as SIFT [13] and RootSIFT [14], while other methods [4,15,16] exploit Convolutional Neural Network (CNN) features and achieve better performance. However, most existing CNN based methods directly use the CNN models that pre-trained on the classification task to extract features, which limits the performance of the pre-trained models in the object instance search task. Some end-to-end object instance search methods adopt the unsupervised training by introducing extra training data [17]. However, these CNN based methods have not taken full advantage of the underlying supervision information of the data. Thus, how to design an effective model for the robust instance search in an unsupervised fashion is still a challenging issue.

To solve this problem, we propose to use the product quantization to generate effective supervision information for unsupervised model learning based on two observations. First, the product quantization methods [18–20] can provide guidance for the search model. In these methods, the feature space is divided into a set of subspaces and each feature is represented by a Cartesian product of several codewords in the subspace. The similar features are grouped into the same cluster and assigned the same codeword indices. The indices actually can be seen as pseudo labels for the model training, which inspires us to explore the practicability of utilizing the product quantization to generate supervision information. Second, the quality of the product quantization largely depends on the quantizability of the input features. The input features exhibiting a cluster structure can be quantized accurately [21].

* Corresponding author.
E-mail addresses: jiangwei33@bit.edu.cn (W. Jiang), wuyuwei@bit.edu.cn (Y. Wu), jingchenchen1996@bit.edu.cn (C. Jing), tyu008@e.ntu.edu.sg (T. Yu), jiayunde@bit.edu.cn (Y. Jia).

https://doi.org/10.1016/j.neucom.2019.06.088
0925-2312/

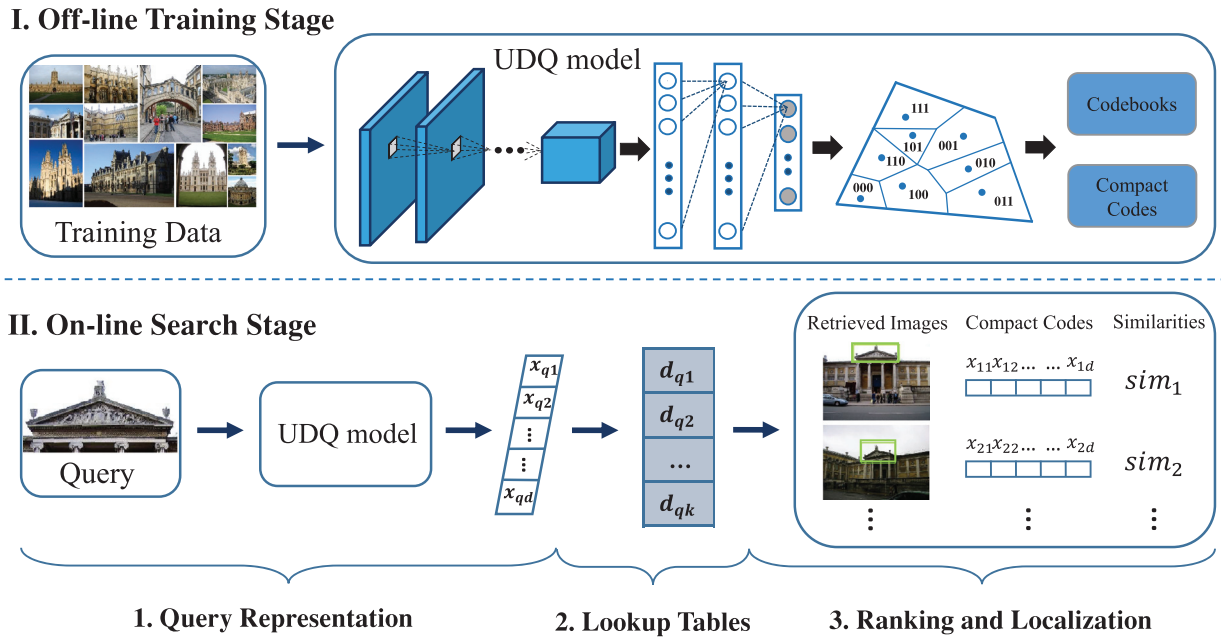


Fig. 1. An overview of our object instance search process. In the off-line training stage, the UDQ model is optimized through a two-module training procedure and generates codebooks and compact codes from the training data. In the on-line search stage, the query object is first input to the trained UDQ model to generate its feature and the distance lookup tables. The search results are finally obtained by similarity ranking according to the lookup tables and the compact codes.

Motivated by these observations, we present an unsupervised deep quantization (UDQ) method for object instance search. Fig. 1 depicts the framework of our method: the off-line training stage and the on-line search stage. The first stage trains a UDQ model and generates both codebooks and the compact code for each proposal that is generated from images. The second stage utilizes the trained UDQ to generate the representation of the query and conducts the efficient search based on distance look-up tables. Fig. 2 illustrates the training procedure of the UDQ. It alternately optimizes the features and makes them exhibit a cluster structure, and generates more effective supervision information for the subsequent training procedure. The training procedure of the UDQ includes two modules: quantization model initialization and self-supervised model optimization. In the first module, the UDQ generates a set of proposal features and exploits product quantization to generate self-supervision information to initialize the model. The second module iteratively carries out two steps: self-supervision information updating and feature optimization until the stopping criterion is met.

In order to encourage the features to satisfy a cluster structure for product quantization, we introduce two constraints, the separability constraint and the discriminability constraint, in the training stage. Meanwhile, the iterative optimization strategy enables the features and the supervision information to be optimized in a unified model, and they alternately enhance each other in an unsupervised manner. In this way, the UDQ is optimized with more effective supervision information and is able to generate more effective and discriminative features. Furthermore, to better guide the model training, we develop three refinement strategies to enrich each feature with its similar features and further discover better supervision information.

After the training stage, the codebooks and compact codes of the reference images are generated. Thanks to the introduction of the product quantization, the UDQ possesses high search efficiency in both time and memory. The entire search procedure is shown in Fig. 1. For a given query object, the UDQ generates its feature and the corresponding distance lookup tables. The search results are obtained by similarity computing and ranking according to the lookup tables and the compact codes. The complexity of the similarity computation is $\mathcal{O}(1)$ by looking up for the distance tables. To evaluate the effectiveness of the UDQ method, we conduct a series of experiments on four widely used object instance search datasets. Experimental results show the approving capability of the UDQ for object instance search.

In summary, our contributions are two-fold.

- We propose an effective unsupervised deep quantization method that iteratively utilizes product quantization to generate self-supervision information and exploits the self-supervision information to guide the model learning in an unsupervised fashion. To the best of our knowledge, the UDQ is the first study of unsupervised deep quantization for object instance search.
- We develop three feature refinement strategies to obtain better supervision information for the model training. Each feature is enriched by fusing with its top-ranked similar features to generate a more discriminative one for better supervision information generation.

2. Related work

Existing work for object instance search can be roughly divided into two categories: methods based on the hand-crafted features and methods based on the CNN features. The methods based on hand-crafted features [10–12] extract local features, such as the SIFT [13] and RootSIFT [14], and learn a codebook off-line. They encode each local feature to its nearest codeword or the combination of some codewords by bag-of-words(BoW) [22], vector of locally aggregated descriptors (VLAD) [23], or fisher vector (FV) [24], *etc.* In addition, the inverted index and binary signatures can be adopted for fast search [25–27]. For instance, Zhang et al. [11] first extracted the SIFT features and built a visual vocabulary tree [28] by hierarchical K-means clustering of the SIFT features. Then they leveraged semantic attributes to update the inverted indexes obtained by the visual vocabulary tree and achieved more satisfactory retrieval results.

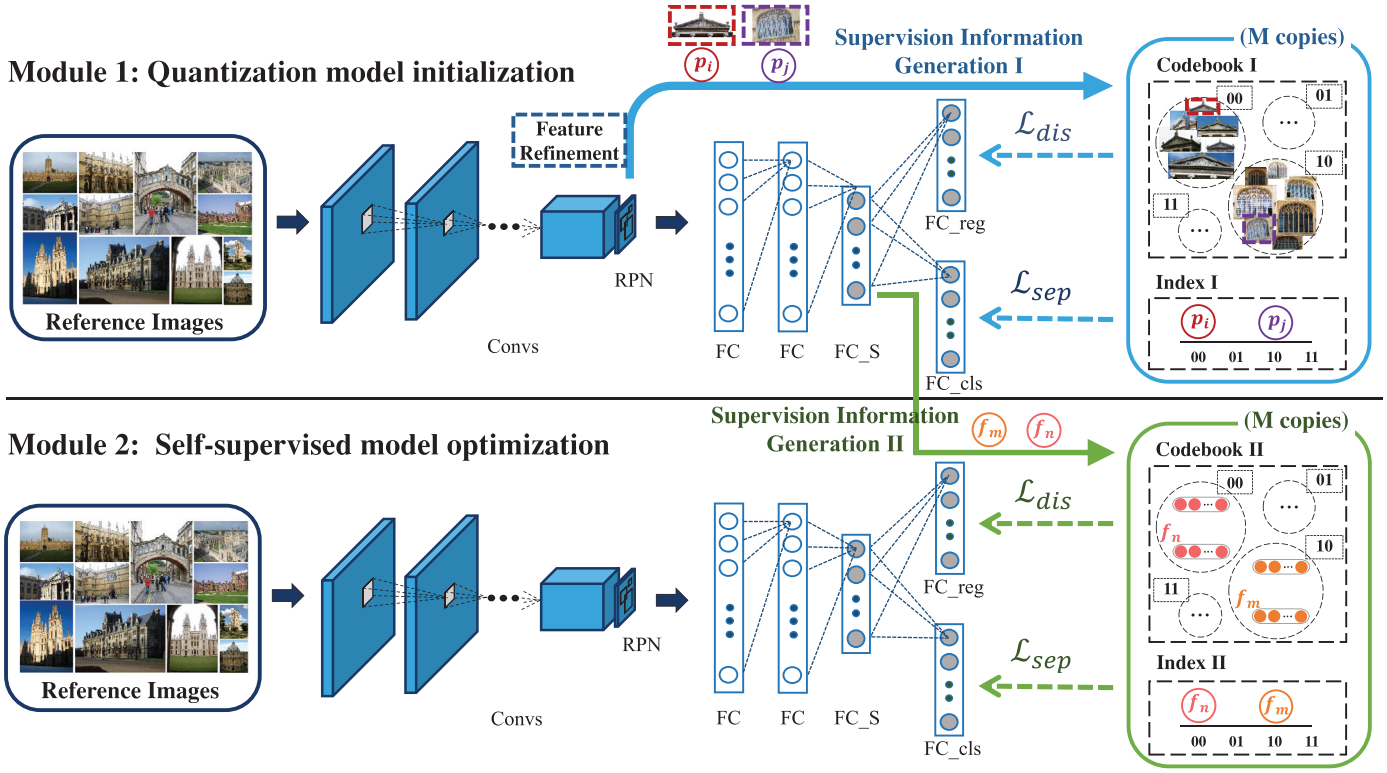


Fig. 2. Our method follows a two-step training procedure. In the first step(top row), the UDQ first generates codebooks and indices (blue solid arrow) according to the output of RPN and then exploits the codebooks and indices as supervision information to initialize the network with the separable loss and the discriminative loss ($\mathcal{L}_{dis}$ and $\mathcal{L}_{sep}$ in blue dash arrow). With the initialized network, in the second step(bottom row), the UDQ iteratively generates codebooks and indices (green solid arrow) according to the output of the "FC_S" layer and further optimizes the network ($\mathcal{L}_{dis}$ and $\mathcal{L}_{sep}$ in green dash arrow) until the stopping criterion is satisfied. Best viewed in colour.

Qin et al. [10] proposed a probabilistic framework for modeling the similarity of two local features such as the SIFT. In this way, a query adaptive distance measurement can be adopted to derive a global similarity between two images. Shi et al. [12] proposed to early detect the visual bursts in an image off-line, and the descriptors were quantized, encoded and searched as usual. Their method promoted the performance of the state-of-the-art image search methods such as VLAD and the selective match kernel [29].

In recent years, the hand-crafted feature seems to be overtaken by the convolutional neural network (CNN) in many vision tasks [1–3,30–34] due to the powerful abilities on learning discriminative high-level representations. In the object instance search, more and more methods based on deep neural networks are proposed with significant performance improvements than traditional hand-crafted feature based object instance search methods. The methods based on the CNN features can be further divided into two categories: the two-step method and the end-to-end method. The two-step method [4,35] first uses the pre-trained CNN models to extract features and then encode the features for object instance search. The CNN models are based on different network architectures, including AlexNet [36], VGGNet [37], GoogleNet [38] and ResNet [39], *etc*. Moreover, different layers of the deep network always exhibit different retrieval performance, for example, the top layers may exhibit lower generalization ability than the layer before it [40]. In particular, Zheng et al. [41] and Tolias et al. [4] proved that the visual representations extracted from the activations of the last convolutional layer perform better than the representations extracted from the fully connected layers, as they convey the image spatial information. With this idea, Tolias et al. [4] proposed a compact image representation that was derived from the convolutional layer activations and encoded multiple image regions. Razavian et al. [42] exploited multi-scale schemes to extract local features and demonstrated the superiority of the image representations that are generated from the generic convolutional networks. Kalantidis et al. [43] proposed an image representation generation method by cross-dimensional weighting and aggregation of convolutional layer activations. Jimenez et al. [15] proposed a local-aware encoding method based on semantic information predicted in the target images and achieved state-of-the-art performance.

The end-to-end methods integrate the feature extraction and encoding into a unified framework to achieve better performance. For instance, Babenko et al. [17] designed a deep convolutional network. It proves the utilization of the image representations that are derived from the top layer of a convolutional network can provide high-level descriptors of the images. Their method also shows that the retrieval performance can be improved by re-training the network with a dataset that is similar to the retrieved images. Salvador et al. [5] took advantage of the fine-tuned Faster R-CNN network [44] for end-to-end representation generation. It demonstrates that fine-tuning an object detector on the target dataset indeed improves the instance search accuracy. Radenovic et al. [16] built a three-stream Siamese network based on the R-MAC method [4] to learn features for object instance search. In addition, Song et al. [7] proposed a deep region hashing method for object instance search. Their method avoids matching a query and the reference images with real-valued features by utilizing hashing to generate binary codes, which significantly improves the search efficiency. Different from these methods, we utilize the product quantization to generate self-supervision information and optimize the self-supervision information through an iterative optimization strategy in an unsupervised fashion.

Table 1
The list of the notations used throughout the paper.

Symbol	Definition
N	The number of reference images in the dataset
T	The number of proposals generated from an image
G	The number of proposals generated from the image dataset
M	The number of subspaces in the product quantizationv
K	The number of codewords in each subspace
S	The number of similar proposals in the refinement strategies
d	The dimension of a feature
d_m	The dimension of a sub-feature in the mth subspace
$\mathcal{X} = \{x_i\}_{i=1}^N$	The set of reference images
$\mathcal{P} = \{p_t\}_{t=1}^G$	The set of proposals
$\mathcal{F} = \{\boldsymbol{f}_t\}_{t=1}^G$	The feature set of the proposals
$\mathcal{H} = \{\boldsymbol{h}_t\}_{t=1}^G$	The compact code set of the proposals
$\widetilde{\mathcal{F}} = \{\widetilde{\boldsymbol{f}}_t\}_{t=1}^G$	The set of quantized features
$\mathcal{Y} = \{\boldsymbol{y}_t\}_{t=1}^G$	The set of one-hot labels
$\mathcal{C}$	The codebook as shown in Eq. (6)
q	A query object
h_t^m	The codeword index of the tth proposal in the mth subspace
$\boldsymbol{f}_t \in \mathbb{R}^d$	The feature of a proposal
$\boldsymbol{c}_t \in \mathbb{R}^d$	The feature of the corresponding codeword
$\boldsymbol{h}_t \in \mathbb{R}^M$	The compact code of a proposal
$\widetilde{\boldsymbol{f}}_t \in \mathbb{R}^d$	The quantized feature of a proposal
$\boldsymbol{y}_t^m \in \mathbb{R}^k$	The one-hot label of a proposal

3. Unsupervised deep quantization method

3.1. Problem formulation

Object instance search aims to retrieve all the reference images containing the specified query object and localize it in these reference images. We list the notations and the corresponding definitions in Table 1. Suppose that we have N reference images $\mathcal{X} = \{x_i\}_{i=1}^N$ and each image x_i has T proposals $\mathcal{P}_i = \{p_{i,j}\}_{j=1}^T$. The similarity between a query q and a reference image x_i is defined by the similarity between the query and the best-matched proposal p_i^* in $\mathcal{P}_i$, which is given by

$$Sim(q, x_i) = \max_j Sim(q, p_{i,j}). \tag{1}$$

$Sim(q, p_{i,j})$ is expressed as

$$Sim(q, p_{i,j}) = \frac{1}{\left\| \boldsymbol{f}_q - \boldsymbol{f}_{p_{i,j}} \right\|_2}, \tag{2}$$

where $\boldsymbol{f}_q$ and $\boldsymbol{f}_{p_{i,j}}$ denote the features of the query q and a proposal $p_{i,j}$ of the reference image x_i, respectively.

In the quantization based object instance search, the feature space is decomposed into the Cartesian product of low-dimensional subspaces and each feature $\boldsymbol{f}$ is divided into M sub-features as $\boldsymbol{f} = [\boldsymbol{f}^1, \boldsymbol{f}^2, \ldots, \boldsymbol{f}^M] \in \mathbb{R}^d$. Each sub-feature is quantized into a codeword and the feature $\boldsymbol{f}$ is approximated by the quantizer function $\phi(\cdot)$ defined as

$$\phi(\boldsymbol{f}) = [\boldsymbol{c}_{h^1}^1, \boldsymbol{c}_{h^2}^2, \ldots, \boldsymbol{c}_{h^M}^M], \tag{3}$$

and h^m is defined as

$$h^m = \arg\min_k \|\boldsymbol{c}_k^m - \boldsymbol{f}^m\|_2, \tag{4}$$

where $\boldsymbol{c}_k^m$ $(m = 1, 2, \ldots, M, k = 1, 2, \ldots, K)$ denotes the kth codeword from the mth codebook, M is the number of subspaces, and K represents the number of codewords in each subspace.

After that, benefiting from the asymmetric distance computation method [18], the similarity between the query q and the reference image x_i is approximated by

$$Sim(q, x_i) = \frac{1}{\left\| \boldsymbol{f}_q - \phi(\boldsymbol{f}_{p_i^*}) \right\|_2}. \tag{5}$$

To simplify the formulation, we define $G = N \times T$ is the number of the proposals generated from the N reference images, thus the proposal set can be rewritten as $\mathcal{P} = \{p_t\}_{t=1}^G$. We further introduce $\mathcal{F} = \{\boldsymbol{f}_t\}_{t=1}^G$ and $\mathcal{H} = \{\boldsymbol{h}_t\}_{t=1}^G$ to denote the features and indices of the proposals, respectively. Under this circumstance, the key problem of achieving effective object instance search performance is how to jointly optimize the proposal features $\mathcal{F}$ and the quantizer which can be represented by the learning of codebooks $\mathcal{C}$ and indices $\mathcal{H}$ in an unsupervised manner.

3.2. Self-supervision information generation

The object instance search is essentially an unsupervised task. Therefore, it is unrealistic to provide enough well-labeled training data for network learning, which motivates us to discover the underlying self-supervision information from the training data to train the UDQ model.

Given a set of features $\mathcal{F} = \{\boldsymbol{f}_t\}_{t=1}^G$, we exploit the optimized product quantization [19] to generate codebooks and indices of the features. The key idea is to minimize the quantization error $\mathcal{L}_{pq}(\mathcal{F})$, given by

$$\begin{aligned} &\mathcal{L}_{pq}(\mathcal{F}) = \sum_{t=1}^G \|\boldsymbol{f}_t - \boldsymbol{c}_t\|_2^2, \\ &s.t. \quad \boldsymbol{c}_t \in \mathcal{C} = \{\boldsymbol{c} | R\boldsymbol{c} \in \mathcal{C}^1 \times \mathcal{C}^2 \times \cdots \times \mathcal{C}^M, R^T R = I\}, \end{aligned} \tag{6}$$

where $\|\cdot\|_2$ denotes the l_2-norm, $\mathcal{C}^m$ is the codebook of the mth subspace, R denotes a rotation matrix and I refers to an identity matrix.

By minimizing the quantization error, the compact codes $\mathcal{H} = \{\boldsymbol{h}_t\}_{t=1}^G$ and the codewords $\mathcal{C}$ are generated at the same time, where $\boldsymbol{h}_t = [h_t^1, h_t^2, \ldots, h_t^M] \in \mathbb{R}^M$. The quantized features $\widetilde{\mathcal{F}} = \{\widetilde{\boldsymbol{f}}_t\}_{t=1}^G$ can be calculated by $\widetilde{\boldsymbol{f}}_t = \phi(\boldsymbol{f}_t)$ according to Eq. (3). We assign a one-hot label $\boldsymbol{y}_t^m$ for each sub-feature $\boldsymbol{f}_t^m$ according to the h_t^m to present the codeword index of $\boldsymbol{f}_t^m$ in the mth subspace. Specifically, the h_t^mth bit of $\boldsymbol{y}_t^m$ is 1 and the others of $\boldsymbol{y}_t^m$ are 0. The $\mathcal{Y} = \{\boldsymbol{y}_t\}_{t=1}^G$ and the $\widetilde{\mathcal{F}} = \{\widetilde{\boldsymbol{f}}_t\}_{t=1}^G$ are utilized as self-supervision information to optimize the UDQ model in an unsupervised manner.

3.3. Model optimization

Fig. 2 shows the architecture of the UDQ model which can be built on different backbone networks. The proposed UDQ consists of a set of convolutional layers, a Region Proposal Network (RPN) [44], a region of interest (RoI) pooling layer, and a set of fully-connected layers. Then the FC_cls layer and the FC_reg layer are utilized to conduct two constraints. Since the features extracted from the RPN may not be readily applied to generate efficient representations for object instance search, we devise a two-module training procedure to alternately update the features and the supervision information in a unified model as shown in Fig. 2.

In the first module, the UDQ generates self-supervision information by minimizing the quantization loss $\mathcal{L}_{pq}$ of the RPN output according to Eq. (6). We introduce the separability constraint $\mathcal{L}_{sep}$ to separate the dissimilar features into different clusters, and the discriminability constraint $\mathcal{L}_{dis}$ to pull the similar features in the same cluster closer to their center. Therefore, the UDQ model is optimized with a joint loss function defined as

$$\mathcal{L} = \mathcal{L}_{sep} + \lambda \mathcal{L}_{dis}, \tag{7}$$

where λ is the hyper-parameter to balance the two supervision signals. $\mathcal{L}_{sep}$ and $\mathcal{L}_{dis}$ are formulated as

$$\mathcal{L}_{sep} = -\sum_{m=1}^M \sum_{t=1}^G \log \frac{e^{W_{m,h_t^m}^T \boldsymbol{f}_t^m + b_{m,h_t^m}}}{\sum_{l=1}^K e^{W_{m,l}^T \boldsymbol{f}_t^m + b_{m,l}}}, \tag{8}$$

$$\mathcal{L}_{dis} = \sum_{m=1}^{M}\sum_{t=1}^{G}\left\| \boldsymbol{f}_t^m - \widetilde{\boldsymbol{f}}_t^m \right\|_2^2, \tag{9}$$

where $\|\cdot\|_2$ denotes the l_2-norm. M represents the number of subspaces and K is the number of codewords in each subspace. $\boldsymbol{f}_t^m \in \mathbb{R}^{d_m}$ denotes the mth part of $\boldsymbol{f}_t^m$ in the mth subspace. h_t^m denotes the codeword index of $\boldsymbol{f}_t^m$. $W_m \in \mathbb{R}^{d_m \times K}$ is weights and $b_m \in \mathbb{R}^K$ is the bias term. $\widetilde{\boldsymbol{f}}_t^m \in \mathbb{R}^{d_m}$ denotes the quantized $\boldsymbol{f}_t^m$ in the mth subspace.

In the second module, the UDQ first generates self-supervision information by minimizing the $\mathcal{L}_{pq}$ of the FC_S layer output according to Eq. (6). Then we exploit the joint loss function according to Eq. (7) to optimize the features with the updated supervision information. The second module iteratively conducts the supervision information updating and the feature optimization until the stopping criterion is satisfied.

With the two constraints, the features are trained to be more discriminant and encouraged to satisfy a cluster structure for the product quantization. Thus, the UDQ can learn better quantizers for the subsequence supervision information generation. The iterative optimization strategy iteratively optimizes the features and the supervision, and leads to more effective supervision information for the model training. In addition, thanks to the introduction of the product quantization, the UDQ can generate nearly cost-free compact representations.

As a result, the final features $\mathcal{F}$, the codebooks $\mathcal{C}$ and compact representations $\mathcal{H}$ for the reference images $\mathcal{X} = \{x_i\}_{i=1}^N$ are obtained by solving

$$\Psi(\mathcal{X}, \Theta) = \{\mathcal{F}, \mathcal{C}, \mathcal{H}\}, \tag{10}$$

where $\Psi(\cdot)$ denotes the forward propagation process and Θ denotes the parameters of the UDQ. Overall, the training procedure of the UDQ is summarized in Algorithm 1.

Algorithm 1: Unsupervised Deep Quantization.

Input: Reference images $\mathcal{X} = \{x_i\}_{i=1}^N$; iteration times *max*.
Output: Final features $\mathcal{F}_{max}$; codebooks $\mathcal{C}$; compact codes $\mathcal{H}$.

```
Generate the proposal features F_0 = {f_t}_{t=1}^G from the RPN;
Generate the initial self-supervision information Y_0 and F̃_0 by minimizing L_pq in Eq. (6) using F_0;
Initialize the deep quantization model parameters Θ with loss function L in Eq. (7) with respect to the Y_0 and F̃_0;
for i = 1 : max do
    Generate the self-supervision information Y_i and F̃_i by minimizing L_pq in Eq. (6) using the output F_i of the FC_S layer;
    Update the deep quantization model parameters Θ by loss function L in Eq. (7) with respect to Y_i and F̃_i;
end
Generate the final features F_max, the final codebooks C and the compact codes H;
Return F_max, C, H.
```

3.4. Feature refinement

Traditional re-ranking methods, such as average query expansion (AQE) [45], have shown their superiority in improving the search results. With the re-ranking strategies, the initial search results are refined by using the top-ranked retrieved images to fuse and create a new query. In this way, the new ranking results are obtained with a kind of blind relevance feedback and lead to a higher recall. Inspired by this, we develop three feature refinement strategies to discover better supervision information. For a feature $\boldsymbol{f}_t \in \mathbb{R}^d$ $(t = 1, 2, \ldots, G)$ in $\mathcal{F}$, the refinement strategies are presented as follows.

- Brute-Force Refinement (BFR). We treat each $\boldsymbol{f}_t$ as a query to perform a whole proposal set matching and ranking process. In this way, the ranking results will introduce relevance feedback of the similar proposals to refine each feature. We use $\boldsymbol{f}_{P_s}$ to denote the feature of $\boldsymbol{f}_t$'s sth nearest object proposal P_s. The BFR refines a proposal's feature $\boldsymbol{f}_t$ by its top-S similar proposals in the proposal set, which is given by

$$\widehat{\boldsymbol{f}}_t = \frac{\boldsymbol{f}_t + \sum_{s=1}^{S} \boldsymbol{f}_{P_s}}{S+1}. \tag{11}$$

- Proposal Based Refinement (PBR). We first perform the k-means method on the proposal set, and encode each $\boldsymbol{f}_t$ to its nearest codeword C_{f_t}. Given an object proposal's feature $\boldsymbol{f}_t$, the PBR only takes object proposals assigned to the same codeword C_{f_t} into consideration. We use $\boldsymbol{f}_{P_s^*}$ to denote the feature of $\boldsymbol{f}_t$'s sth nearest object proposal among the proposals assigned to C_{f_t}. PBR refines $\boldsymbol{f}_t$ by its top-S similar object proposal assigned to the same codeword C_{f_t}, which is given by

$$\widehat{\boldsymbol{f}}_t = \frac{\boldsymbol{f}_t + \sum_{s=1}^{S} \boldsymbol{f}_{P_s^*}}{S+1}. \tag{12}$$

- Center Based Refinement (CBR). Similar to the PBR, each $\boldsymbol{f}_t$ in CBR is first encoded to its nearest codeword C_{f_t} by the k-means. Then $\widehat{\boldsymbol{f}}_t$ is directly calculated by

$$\widehat{\boldsymbol{f}}_t = \frac{\boldsymbol{f}_t + \boldsymbol{f}_{C_{f_t}}}{2}, \tag{13}$$

  where $\boldsymbol{f}_{C_{f_t}}$ presents the feature of $\boldsymbol{C}_{f_t}$.

With the refinement strategies, each feature is enriched with similar features. In this way, the supervision information generated from the refined features is more effective, which leads to better model optimization with multiple iterations. The experiments validate the effectiveness of the feature refinement.

Among the three refinement strategies, the BFR is supposed to be the most effective since the exhaustive searching and ranking process guarantees the most relevant features are used to refine the original features. In contrast, the PBR and the CBR significantly improve the efficiency of the whole training stage by exploiting the features that belong to the same cluster for the refinement.

3.5. Object instance search

In the object instance search task, we need to compute the distance between the query representation and each reference image representation to determine whether they are similar. The entire search procedure is shown in Fig. 1. After the representation of the query q generated from the trained UDQ, the distance lookup tables are first built based on the distance between the query and the codewords. Then the similarity between the query q and the reference image x_i can be obtained by looking up the distance tables according to Eq. (5). Finally, the similarity between the query and all reference images is computed and ranked to obtain the search results, and the position of the best similar proposal in each reference image indicates the position of the query object in that image.

Taking advantage of the product quantization, the UDQ can obtain nearly cost-free codebooks and compact codes from the model. Thus, it only takes $M\log_2(K)$ bits to store a compact representation $\boldsymbol{h}_{p_t}$ for the proposal p_t, where M and K are the subspace number and the codeword number in each subspace, respectively. Moreover, benefiting from the lookup tables, the computational complexity of computing the distance between a query

and a proposal is $\mathcal{O}(1)$. As a result, the UDQ significantly boosts the efficiency in both time and memory.

4. Experiments

We provide a comprehensive comparison with various baselines and state-of-the-art methods on four object instance search datasets, the Oxford5K dataset [46], the Oxford105K dataset [46], the Paris6K dataset [47] and the Paris106K dataset [47].

4.1. Datasets

The Oxford5K dataset [46] contains 5063 images that download from Flickr and each image is named by a landmark in Oxford. It has 55 query images in 11 classes and each query is an object image cut from one big image with its ground truth bounding box. We further add 100K distractor images called Flickr100K [47] that download from Flickr to compose the Oxford105K dataset for testing the scalability of our method on the large-scale image search scenario.

The Paris6K dataset [47] contains 6392 images. Each of them is download from Flickr and describes a specific Paris architecture. Since each landmark has 5 queries, there are 55 query images with bounding boxes as well. Similar to the Oxford105K dataset, we compose the Paris106K by adding the Flickr100K dataset to the Paris6K dataset.

To better localize the object in the images, we follow the Ren et al. [44] to pre-train the RPN on the PASCAL VOC 2007 dataset [48]. It is a widely used dataset for object detection and contains 9963 images of 20 classes including animals, person, vehicle and indoor objects. This dataset consists of about 5K trainval images and 5K test images. We train the RPN on the VOC 2007 trainval images.

4.2. Experiment setting

The UDQ is built on different backbone networks, we utilize two widely used networks VGG16 [37] and ResNet101 [39] as our backbones to make fair comparisons with the existing methods adopting the same architectures. The VGG16-based UDQ consists of 5 groups of convolutional layers and 4 max-pooling layers. The ResNet101-based UDQ consists of 4 building blocks and each building block is comprised from a set of convolutional layers. Both of the networks are followed by a RPN, a ROI pooling layer, and three fully-connected layers.

The backbone networks are pre-trained on the ImageNet dataset to initialize the convolutional layers. Then we fix the backbone network and train the RPN on PASCAL VOC dataset [49]. Following Salvador et al. [5], we reshape the whole query images so that their shorter sides are 600 pixels while their longer size after the reshape are smaller than 1000 pixels. The reshaped query images are used for fine-tuning all the convolutional layers except for the first two convolutional groups with a learning rate of 0.001. In the off-line training stage, we perform the PCA to reduce the dimension of features to 300 before the supervision information generation. The learning rate for the UDQ training is 0.001 as well. In the on-line search stage, the mean Average Precision (mAP) is adopted as the evaluation metric to comprehensively evaluate the performance. For each reference image, 200 proposals are generated.

4.3. Results and discussion

4.3.1. The effect of the training strategies

We devise a two-module training stage as shown in Fig. 2. The first module is an initialization module and the second module is an iterative updating module. We first evaluate the effect of the iteration times in the second module by comparing the proposed UDQ with two methods: “Off-the-shelf” and the “Off-the-shelf-OPQ”. The former one directly uses the off-the-shelf features from the fine-tuned ROI feature maps and the latter further conducts optimized product quantization (OPQ) on the off-the-shelf features from the ROI feature maps. The UDQ is first initialized with the quantization model initialization module, and the search results about the iteration times of the second module are shown in Fig. 3. We conduct experiments on the VGG16-based UDQ and the ResNet101-based UDQ on Oxford5K and Paris6K dataset. In all cases, the subspace number is fixed as 4 and the codeword number in each subspace is fixed as 256.

From Fig. 3, not surprisingly, the performance of the compact “Off-the-shelf-OPQ” representations is worse than the real-valued “Off-the-shelf” representations. This is mainly caused by the information loss brought by the quantization. When the UDQ is only trained with the quantization model initialization module, the results perform worse than “Off-the-shelf-OPQ”. The main reason is that a single iteration of the self-supervised optimization is insufficient to make full use of the underlying information of the data. When the iteration is 1, the UDQ achieves more than 10% improvement on both datasets. Interestingly, we find the performance of the proposed UDQ saturates when the iteration increases beyond 2. Taking both effectiveness and efficiency into consideration, we choose to iterate 2 times in the second module by default.

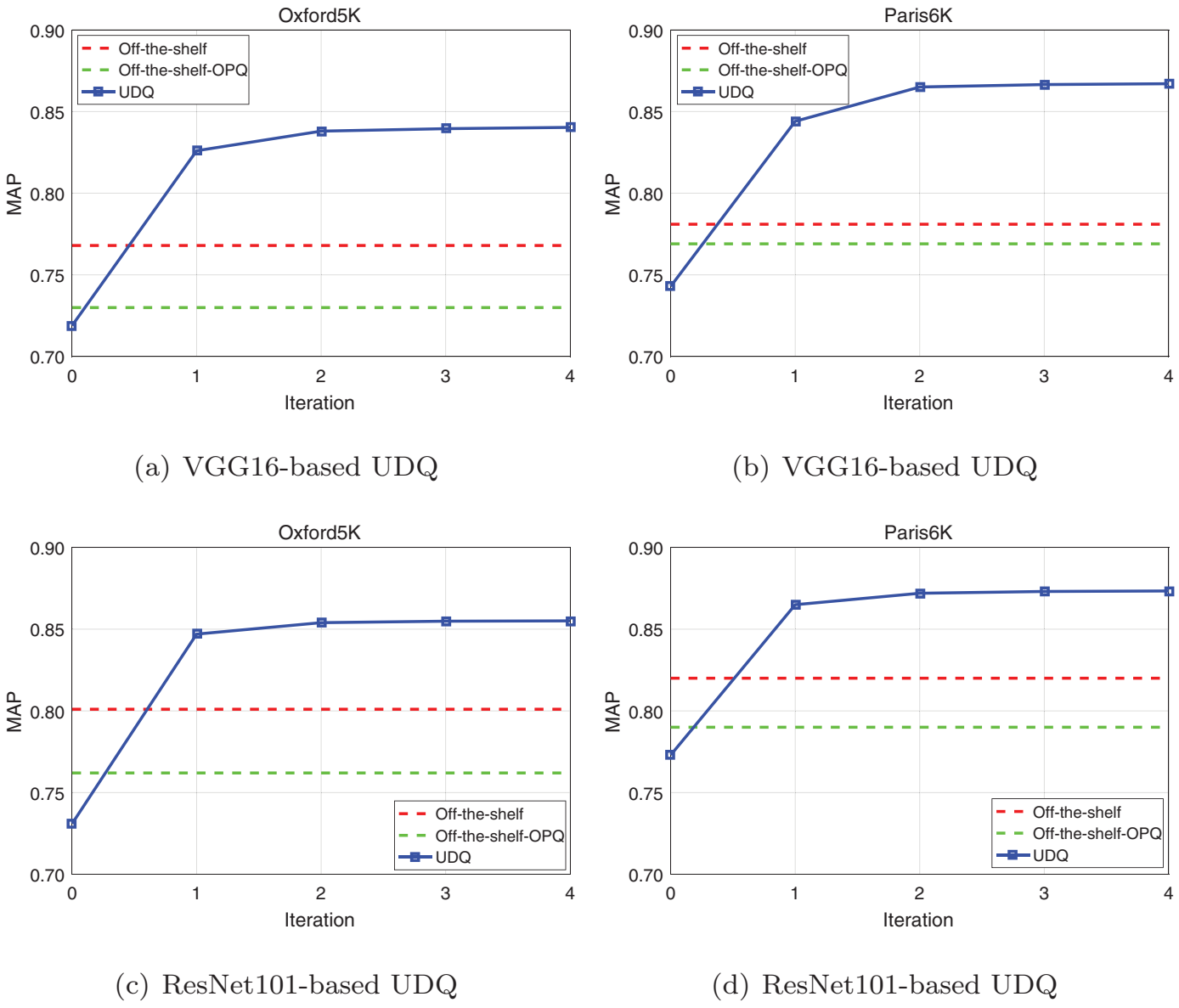
To further validate the effectiveness of the UDQ training strategies, we compare the off-the-shelf-OPQ results and the UDQ results when using different subspace numbers and sub-center numbers. The results based on the VGG16 model and the Oxford5K dataset are shown in Table 2. We observe that the UDQ significantly promotes the search performance, which proves that it indeed generates effective supervision information for the model training.

4.3.2. The effect of the hyper-parameters

In this section, we evaluate the influence of the hyper-parameters in this paper. The λ is used to balance the importance of the two constraints. The subspace subspace number M and the codeword number K in each subspace dominate the effectiveness of the supervision information and the dimension of the compact codes generated from the UDQ together. All of them are essential to our method. So we conduct two experiments to investigate the sensitiveness of these parameters.

First, we fix the subspace number as 4 and the codeword number in each subspace as 256 to validate the sensitive of the λ. The results are shown in Fig. 4. We vary λ among $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$ and conduct experiments on both the Oxford5K dataset and the Paris6K dataset with the VGG16-based UDQ model. We observe that the performance of the UDQ largely depends on the hyper-parameter λ. The best performance is achieved when λ is set as 0.01 in both of the two datasets, in which case the two losses are roughly equally weighted. It should be noted that we did not conduct experiments with larger λ, since the UDQ is difficult to be trained when the λ is larger than 0.01. This is mainly because the $\mathcal{L}_{dis}$ dominates the training. As aforementioned, the $\mathcal{L}_{sep}$ is used to separate the dissimilar features into different clusters, and $\mathcal{L}_{dis}$ further encourages the similar features closer to their center. In other words, the $\mathcal{L}_{dis}$ depends on the $\mathcal{L}_{sep}$. As a result, we set λ as 0.01 in the following experiments.

Then we vary M among {2, 4, 8}, and vary K among {64, 128, 256, 512}. The results are shown in Tables 3 and 4, and we highlight the best results in each column. We observe that the results based on the ResNet101 always better than the results based on the VGG16, which is reasonable due to the more general and



(a) VGG16-based UDQ

(b) VGG16-based UDQ

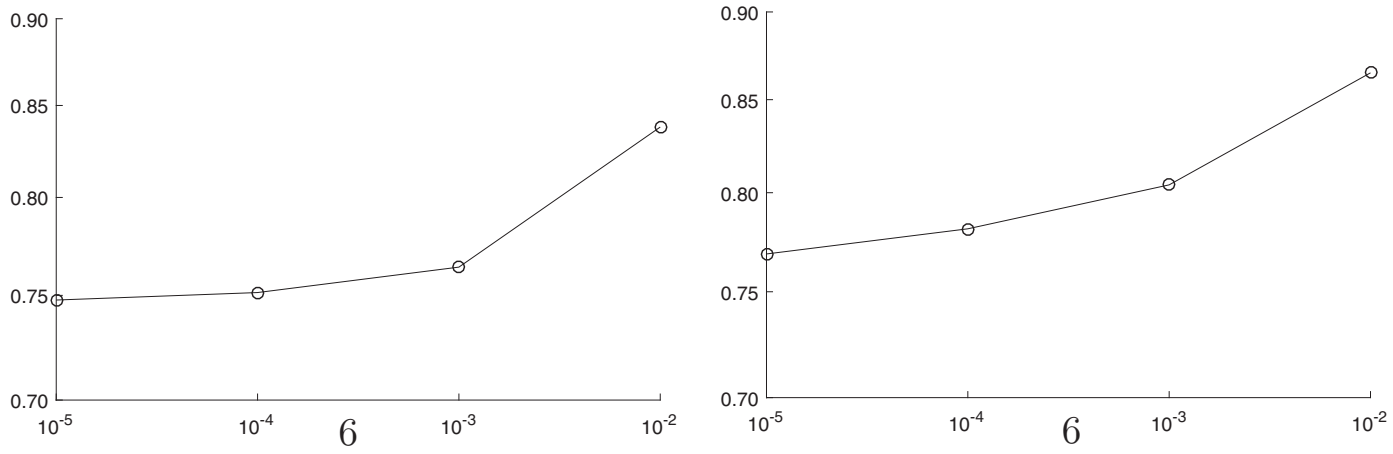
(c) ResNet101-based UDQ

(d) ResNet101-based UDQ

Fig. 3. The influence of the iteration times in the second training module using the VGG16-based UDQ model and the ResNet101-based UDQ model on the Oxford5K dataset and the Paris6K dataset.

Table 2
The influence of the training strategies on the Oxford5K dataset using the VGG-based model.

Subspace numbers	Compact codes	Off-the-shelf-OPQ results	UDQ results
M=2	12bit	0.494	0.514
	14bit	0.682	0.697
	16bit	0.746	0.771
	18bit	0.779	0.815
M=4	24bit	0.633	0.673
	28bit	0.687	0.743
	32bit	0.730	0.838
	36bit	0.781	0.868
M=8	48bit	0.640	0.744
	56bit	0.689	0.801
	64bit	0.743	0.855
	72bit	0.766	0.873



(a) Oxford5K (b) Paris6K

Fig. 4. The influence of the λ in the training stage using the VGG16-based UDQ model on the Oxford5K dataset and the Paris6K dataset.

Table 3
The influence of parameters on the Oxford5K dataset and the Oxford105K dataset using the VGG16-based UDQ model and the ResNet101-based UDQ model.

Subspace number	Compact codes	VGG16-based UDQ		ResNet101-based UDQ	
		Oxford5K	Oxford105K	Oxford5K	Oxford105K
M=2	12bit	0.514	0.491	0.643	0.620
	14bit	0.697	0.679	0.751	0.732
	16bit	0.771	0.755	0.813	0.798
	18bit	0.815	0.805	0.836	0.824
M=4	24bit	0.673	0.661	0.704	0.686
	28bit	0.743	0.728	0.766	0.751
	32bit	0.838	0.829	0.857	0.848
	36bit	0.868	0.861	0.876	0.869
M=8	48bit	0.744	0.734	0.770	0.764
	56bit	0.801	0.794	0.824	0.817
	64bit	0.855	0.849	0.869	0.865
	72bit	**0.873**	**0.870**	**0.887**	**0.883**

Table 4
The influence of parameters on the Paris6K dataset and the Paris106K dataset using the VGG16-based UDQ model and the ResNet101-based UDQ model.

Subspace Number	Compact Codes	VGG16-based UDQ		ResNet101-based UDQ	
		Paris6K	Paris106K	Paris6K	Paris106K
M=2	12bit	0.645	0.621	0.674	0.651
	14bit	0.769	0.748	0.772	0.753
	16bit	0.821	0.807	0.843	0.827
	18bit	0.844	0.839	0.871	0.867
M=4	24bit	0.747	0.733	0.759	0.745
	28bit	0.841	0.835	0.858	0.850
	32bit	**0.865**	**0.857**	0.871	0.859
	36bit	0.863	0.851	0.872	0.863
M=8	48bit	0.818	0.802	0.835	0.821
	56bit	0.854	0.849	0.869	0.862
	64bit	0.863	0.855	**0.877**	**0.871**
	72bit	0.859	0.851	0.875	0.869

powerful ability of the ResNet101 to provide effective features, and it will improve the supervision information quality in the training stage. For the Oxford5K dataset and the Oxford105K dataset, the best results are obtained when $M = 8$ and $K = 512$, which is expected due to two aspects. First, in the off-line training stage, the more subspaces and codewords lead to richer codebooks and less information loss. Thus it provides more effective supervision information for the UDQ model training. Second, in the on-line search stage, better quantization quality and finer cluster partition significantly improve the search results. For the Paris6K dataset and Paris106K dataset, the best results are obtained when $M = 4$ and $K = 256$ based on the VGG16 and $M = 8$ and $K = 256$ based on the ResNet101. This is mainly caused by the over-fitting when both M and K are large. By default, on the following experiments, we set the ResNet101 as the backbone network, the M and K are set as 8 and 512 when testing on the Oxford5K dataset and the Oxford105K dataset, and 8 and 256 when testing on the Paris6K dataset and the Paris106K dataset.

Table 5
Comparison mAPs of the proposed UDQ and the state-of-the-art compact descriptor based methods on four datasets. For each dataset, the two columns show the performance without (W/O) and with (W) post-processing strategies, respectively. We highlight the best result (bold) and the second-best result (italic) on each dataset.

Methods	Oxford5K		Oxford105K		Paris6K		Paris106K	
	W/O	W	W/O	W	W/O	W	W/O	W
Mikulik et al. [50]	0.742	0.849	0.674	0.795	0.749	0.824	0.675	0.773
Qin et al. [10]	0.780	0.850	0.728	0.816	0.736	0.855	N/A	N/A
Zhang et al. [11]	0.687	N/A	0.605	N/A	N/A	N/A	N/A	N/A
Shi et al. [12]	0.813	N/A	0.689	N/A	0.775	N/A	N/A	N/A
Zhang et al. [51]	0.816	N/A	0.761	N/A	0.836	N/A	0.788	N/A
Song et al. [7]	0.783	0.851	N/A	*0.825*	0.801	0.849	N/A	*0.802*
Yang et al. [9]	0.831	*0.851*	0.795	N/A	0.849	*0.872*	0.734	N/A
Do et al. [52]	0.703	0.742	0.447	0.622	N/A	N/A	N/A	N/A
Zhang et al. [53]	0.809	0.836	N/A	N/A	0.796	0.814	N/A	N/A
UDQ-Baseline	0.887		0.883		0.877		0.871	
UDQ-PBR	0.893		0.890		0.885		0.878	
UDQ-CBR	0.891		0.886		0.893		0.889	
UDQ-BFR	**0.901**		**0.897**		**0.900**		**0.893**	

Table 6
Comparison mAPs of the proposed compact descriptor based UDQ and real-valued descriptor based the state-of-the-art methods on four datasets. For each dataset, the two columns show the method performance without (W/O) and with (W) post-processing strategies, respectively. We highlight the best result (bold) on each dataset.

Methods	Code length	Oxford5K		Oxford105K		Paris6K		Paris106K	
		W/O	W	W/O	W	W/O	W	W/O	W
Tolias et al. [4]	512d	0.669	0.773	0.616	0.732	0.830	0.865	0.757	0.798
Razavian et al. [42]	15k/32k	0.655	0.843	0.489	N/A	0.685	0.879	N/A	N/A
Kalantidis et al. [43]	512d	0.708	0.749	0.653	0.706	0.797	0.848	0.722	0.794
Jimenez et al. [15]	512d	0.736	0.811	0.672	0.769	0.855	0.874	0.733	0.800
Radenovic et al. [16]	512d/2048d	0.878	**0.910**	0.846	0.895	0.927	**0.955**	0.869	**0.919**
UDQ-BFR	8bytes/9bytes	0.901		**0.897**		0.900		0.893	

4.3.3. The effect of the feature refinement strategies

We evaluate the feature refinement strategies and the results are shown in Table 5. "UDQ-Baseline" refers to the UDQ model that is trained without any refinement. We choose top-50 and top-10 feedbacks to refine the features in PBR and BFR, respectively.

From Table 5, we have the following observations: (1) All of the feature refinement strategies can significantly improve the performance of the baseline UDQ model, which demonstrates better supervision information has been obtained. (2) The BFR performs best thanks to its exhaustive ranking process. (3) The PBR and the CBR keep satisfied performance while significantly reducing the computational complexity of the BFR. Overall, the feature refinement strategy is able to obtain better supervision information for the model training and it effectively improves the search performance.

4.3.4. Comparisons with the state-of-the-art methods

To evaluate the effectiveness of our method, we compare the UDQ with the state-of-the-art methods on four datasets. We divide the comparison methods into compact descriptor based methods and real-valued descriptor methods. The former indicates the methods that are devoted to fast search with compact representations and the latter indicates the methods that are devoted to satisfied search performance with real-valued representations.

We first compare the UDQ with the compact descriptor based methods. The comparison results of each method without and with post-processing strategies are listed for all the datasets in Table 5. From the results, we can see that the UDQ significantly outperforms all the state-of-the-art methods even though they incorporate various post-processing strategies. It should be noted that, for the most datasets, the baseline UDQ achieves an inspiring accuracy improvement comparing to these state-of-the-art methods. Our best results outperform the second best methods by 5.0%, 7.2%, 2.8%, 9.1% in terms of mAP on the four datasets, respectively.

We further compare the UDQ with the real-valued descriptor based methods. The results are shown in Table 6. The UDQ achieves a slightly lower performance than the method in [16] and the main reasons are two aspects. First, it feeds the images to the network at multi-scale, which indeed enlarges the image dataset. Second, it exploits high-dimensional real-valued features for search while the UDQ is based on the compact codes with information loss. By comparison, our UDQ significantly reduces the time and memory cost for the object instance search. As a result, it achieves a better trade-off of the search efficiency and accuracy. Some qualitative results of our method are shown in Fig. 5.

Note that when adding 100K distractor images, the UDQ remains satisfied performance, which significantly demonstrates the robustness of our model. The main reasons are as follows. (1) The separability constraint and the discriminability constraint enable the features to be more suitable for the effective supervision information generation. (2) The iterative optimization strategy guarantees the features and the supervision information can be enhanced each other alternately in a unified model. (3) The refinement strategy further shows the effectiveness of the supervision information and leads to a more effective search model.

In order to better validate the efficiency of our method, we conduct experiments on the Oxford5K dataset and the Paris6K dataset to evaluate the efficiency of the UDQ. We choose Song et al. [7] as the representative algorithm for hashing methods and Tolias et al. [4] as the representative algorithm for real-valued descriptor based methods. All the experiments are performed on a server with an Intel Core i7-5930 CPU @ 3.50 GHz. The comparison results are shown in Table 7. From the table, we can find that UDQ takes the least search time among methods due to the product quantization.

Fig. 5. Selected results of top-5 ranked images of our method on the Oxford5K, Paris6K, Oxford105K and Paris106K datasets. First image in each row is the query object and the other images are the returned images.

Table 7
Comparison search time (s) on the Oxford5K dataset and the Paris6K dataset.

Datasets	UDQ	Hashing descriptors [7]	Real-valued descriptors [4]
Oxford5K	0.080	0.096	0.169
Paris6K	0.101	0.118	0. 182

As a result, the UDQ model can generate the cost-free codebooks and compact codes and further achieve fast search.

5. Conclusion

In this paper, we have presented an effective Unsupervised Deep Quantization (UDQ) method for object instance search. The proposed UDQ can generate effective supervision information and iteratively optimize the supervision information in an unsupervised fashion. By the introduction of the two constraints, the features are satisfied with a cluster structure and can generate effective supervision information. With the iteration optimization strategy, the UDQ can realize the alternate optimization of the features and the supervision information in a unified model. We further proposed three refinement strategies that can obtain better supervision information. Our UDQ achieves nearly cost-free compact representations and greatly promotes the search efficiency on large-scale datasets. Experimental results show that the UDQ outperforms the state-of-the-art methods.

Declarations of interest

None.

Acknowledgments

This work was supported in part by the Natural Science Foundation of China (NSFC) under Grants Nos. 61702037 and 61773062, Beijing Municipal Natural Science Foundation under Grant no. L172027, and Beijing Institute of Technology Research Fund Program for Young Scholars.

References

[1] G. Ding, J. Zhou, Y. Guo, Z. Lin, S. Zhao, J. Han, Large-scale image retrieval with sparse embedded hashing, Neurocomputing 257 (2017) 24–36.
[2] A. Gordo, J. Almazan, J. Revaud, D. Larlus, End-to-end learning of deep visual representations for image retrieval, Int. J. Comput. Vis. 124 (2) (2017) 237–254.
[3] C. Bai, L. Huang, X. Pan, J. Zheng, S. Chen, Optimization of deep convolutional neural network for large scale image retrieval, Neurocomputing 303 (2018) 60–67.
[4] G. Tolias, R. Sicre, H. Jégou, Particular object retrieval with integral max-pooling of CNN activations, in: Proceedings of the ICML, 2016.
[5] A. Salvador, X. Giró-i Nieto, F. Marqués, S. Satoh, Faster r-CNN features for instance search, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2016, pp. 9–16.
[6] C. Yan, H. Xie, J. Chen, Z. Zha, X. Hao, Y. Zhang, Q. Dai, A fast Uyghur text detector for complex background images, IEEE Trans. Multimed. 20 (12) (2018) 3389–3398.

[7] J. Song, T. He, L. Gao, X. Xu, H.T. Shen, Deep region hashing for generic instance search from images, in: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
[8] T. Yu, Y. Wu, S.D. Bhattacharjee, J. Yuan, Efficient object instance search using fuzzy objects matching., in: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, 2017, pp. 4320–4326.
[9] J. Yang, J. Liang, H. Shen, K. Wang, P.L. Rosin, M.-H. Yang, Dynamic match kernel with deep convolutional features for image retrieval, IEEE Trans. Image Process. 27 (11) (2018) 5288–5302.
[10] D. Qin, C. Wengert, L. Van Gool, Query adaptive similarity for large scale object retrieval, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 1610–1617.
[11] S. Zhang, M. Yang, X. Wang, Y. Lin, Q. Tian, Semantic-aware co-indexing for image retrieval, in: Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 1673–1680.
[12] M. Shi, Y. Avrithis, H. Jégou, Early burst detection for memory-efficient image retrieval, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 605–613.
[13] D.G. Lowe, Distinctive image features from scale-invariant keypoints, Int. J. Comput. Vis. 60 (2) (2004) 91–110.
[14] R. Tao, E. Gavves, C.G. Snoek, A.W. Smeulders, Locality in generic instance search from one example, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 2091–2098.
[15] A. Jimenez, J.M. Alvarez, X. Giro-i Nieto, Class-weighted convolutional features for visual instance search, arXiv:1707.02581(2017).
[16] F. Radenović, G. Tolias, O. Chum, Fine-tuning CNN image retrieval with no human annotation, IEEE Trans. Pattern Anal. Mach. Intell. 41 (7) (2018) 1655–1668.
[17] A. Babenko, A. Slesarev, A. Chigorin, V. Lempitsky, Neural codes for image retrieval, in: Proceedings of the European Conference on Computer Vision, Springer, 2014, pp. 584–599.
[18] H. Jégou, M. Douze, C. Schmid, Product quantization for nearest neighbor search, IEEE Trans. Pattern Anal. Mach. Intell. 33 (1) (2011) 117–128.
[19] T. Ge, K. He, Q. Ke, J. Sun, Optimized product quantization, IEEE Trans. Pattern Anal. Mach. Intell. 36 (4) (2014) 744–755.
[20] T. Yu, J. Yuan, C. Fang, H. Jin, Product quantization network for fast image retrieval, in: Proceedings of the European Conference on Computer Vision, 2018, pp. 186–201.
[21] Y. Cao, M. Long, J. Wang, H. Zhu, Q. Wen, Deep quantization network for efficient image retrieval, in: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, 2016.
[22] J. Sivic, A. Zisserman, Video Google: a text retrieval approach to object matching in videos, in: Proceedings of the NULL, IEEE, 2003, p. 1470.
[23] H. Jégou, M. Douze, C. Schmid, P. Pérez, Aggregating local descriptors into a compact image representation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2010, pp. 3304–3311.
[24] F. Perronnin, J. Sánchez, T. Mensink, Improving the fisher kernel for large-scale image classification, in: Proceedings of the European Conference on Computer Vision, Springer, 2010, pp. 143–156.
[25] F. Perronnin, Y. Liu, J. Sánchez, H. Poirier, Large-scale image retrieval with compressed fisher vectors, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2010, pp. 3384–3391.
[26] Y. Zhang, Z. Jia, T. Chen, Image retrieval with geometry-preserving visual phrases, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2011, pp. 809–816.
[27] Y. Liu, Y. Guo, S. Wu, M.S. Lew, Deepindex for accurate and efficient image retrieval, in: Proceedings of the ACM on International Conference on Multimedia Retrieval, ACM, 2015, pp. 43–50.
[28] D. Nister, H. Stewenius, Scalable recognition with a vocabulary tree, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2, IEEE, 2006, pp. 2161–2168.
[29] G. Tolias, Y. Avrithis, H. Jégou, To aggregate or not to aggregate: selective match kernels for image search, in: Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 1401–1408.
[30] S. Pu, Y. Song, C. Ma, H. Zhang, M.-H. Yang, Deep attentive tracking via reciprocative learning, in: Advances in Neural Information Processing Systems, 2018, pp. 1931–1941.
[31] J. Dong, X. Li, C.G. Snoek, Predicting visual features from text for image and video caption retrieval, IEEE Trans. Multimed. 20 (12) (2018) 3377–3388.
[32] G. Yang, Y. Cui, S. Belongie, B. Hariharan, Learning single-view 3d reconstruction with limited pose supervision, in: Proceedings of the European Conference on Computer Vision, 2018, pp. 86–101.
[33] C. Yan, L. Li, C. Zhang, B. Liu, Y. Zhang, Q. Dai, Cross-modality bridging and knowledge transferring for image understanding, IEEE Trans. Multimed. (2019).
[34] C. Yan, Y. Tu, X. Wang, Y. Zhang, X. Hao, Q. Dai, Stat: spatial-temporal attention mechanism for video captioning, IEEE Trans. Multimed. (2019).
[35] E. Mohedano, K. McGuinness, N.E. O'Connor, A. Salvador, F. Marqués, X. Giro-i Nieto, Bags of local convolutional features for scalable instance search, in: Proceedings of the ACM on International Conference on Multimedia Retrieval, ACM, 2016, pp. 327–331.
[36] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.
[37] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv:1409.1556 (2014).
[38] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.
[39] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[40] H. Azizpour, A.S. Razavian, J. Sullivan, A. Maki, S. Carlsson, Factors of transferability for a generic convnet representation, IEEE Trans. Pattern Anal. Mach. Intell. 38 (9) (2016) 1790–1802.
[41] L. Zheng, Y. Zhao, S. Wang, J. Wang, Q. Tian, Good practice in CNN feature transfer, arXiv:1604.00133 (2016).
[42] A.S. Razavian, J. Sullivan, S. Carlsson, A. Maki, Visual instance retrieval with deep convolutional networks, ITE Trans. Media Technol. Appl. 4 (3) (2016) 251–258.
[43] Y. Kalantidis, C. Mellina, S. Osindero, Cross-dimensional weighting for aggregated deep convolutional features, in: Proceedings of the European Conference on Computer Vision, Springer, 2016, pp. 685–701.
[44] S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: Towards real-time object detection with region proposal networks, in: Advances in Neural Information Processing Systems, 2015, pp. 91–99.
[45] O. Chum, J. Philbin, J. Sivic, M. Isard, A. Zisserman, Total recall: automatic query expansion with a generative feature model for object retrieval, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2007, pp. 1–8.
[46] J. Philbin, O. Chum, M. Isard, J. Sivic, A. Zisserman, Object retrieval with large vocabularies and fast spatial matching, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2007, pp. 1–8.
[47] J. Philbin, O. Chum, M. Isard, J. Sivic, A. Zisserman, Lost in quantization: Improving particular object retrieval in large scale image databases, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2008, pp. 1–8.
[48] M. Everingham, L. Van Gool, C.K. Williams, J. Winn, A. Zisserman, The pascal visual object classes challenge 2007 (voc2007) results, 2007.
[49] M. Everingham, L. Van Gool, C.K. Williams, J. Winn, A. Zisserman, The pascal visual object classes (voc) challenge, Int. J. Comput. Vis. 88 (2) (2010) 303–338.
[50] A. Mikulik, M. Perdoch, O. Chum, J. Matas, Learning vocabularies over a fine quantization, Int. J. Comput. Vis. 103 (1) (2013) 163–175.
[51] G. Zhang, Z. Zeng, S. Zhang, Y. Zhang, W. Wu, Sift matching with CNN evidences for particular object retrieval, Neurocomputing 238 (2017) 399–409.
[52] T.-T. Do, N.-M. Cheung, Embedding based on function approximation for large scale image search, IEEE Trans. Pattern Anal. Mach. Intell. 40 (3) (2018) 626–638.
[53] D. Zhang, J. Tang, G. Jin, Y. Zhang, Q. Tian, Region similarity arrangement for large-scale image retrieval, Neurocomputing 272 (2018) 461–470.

Wei Jiang is currently a master degree candidate in the Beijing Institute of Technology, Beijing, China. Her research interests include image retrieval and video retrieval.

Yuwei Wu received the Ph.D. degree in computer science from Beijing Institute of Technology (BIT), Beijing, China, in 2014. He is now an Assistant Professor at School of Computer Science, BIT. From August 2014 to August 2016, he was a post-doctoral research fellow at Rapid-Rich Object Search (ROSE) Lab, School of Electrical & Electronic Engineering (EEE), Nanyang Technological University (NTU), Singapore. He received outstanding Ph.D. Thesis award from BIT, and Distinguished Dissertation Award Nominee from China Association for Artificial Intelligence (CAAI).

Chenchen Jing received the B.S. degree in computer science in 2016 from Beijing Institute of Technology, Beijing, China, where he is currently a Ph.D. candidate. His research interests include computer vision, machine learning, and multimedia understanding.

Tan Yu is currently pursuing his Ph.D. degree in Electrical and Electronic Department of Nanyang Technological University, Singapore. His research focuses on image and video retrieval.

Yunde Jia received the B.S., M.S., and Ph.D. degrees from the Beijing Institute of Technology (BIT) in 1983, 1986, and 2000, respectively. He was a visiting scientist with the Robotics Institute, Carnegie Mellon University (CMU), from 1995 to 1997. He is currently a Professor with the School of Computer Science, BIT, and the team head of BIT innovation on vision and media computing. He serves as the director of Beijing Lab of Intelligent Information Technology. His interests include computer vision, vision-based HCI and HRI, and intelligent robotics.