Copyright ⓒ2016 KSII

# Enhanced VLAD

**Benchang Wei[1,2], Tao Guan1[1], Yawei Luo[1], Liya Duan[3], Junqing Yu[1]**
[1]School of Computer Science and Technology, Huazhong University of Science & Technology,
Wuhan, China
[e-mail:gt_qd@126.com, royalvane@hust.edu.cn, yjqhust@qq.com]
2School of Electrical and Information Engineer, Hubei University of Automotive Technology
Shiyan, China
[e-mail:bc_david@163.com]
[3]Institute of Oceanographic Instrumentation, Shandong Academy of Sciences
Qingdao, China
[e-mail:liya_duan@126.com]
*Corresponding author: Tao Guan

*Received December 11, 2014; revised May 11, 2015; accepted May 14, 2016; published July 31, 2016*

## Abstract

Recently, Vector of Locally Aggregated Descriptors (VLAD) has been proposed to index image by compact representations, which encodes powerful local descriptors and makes significant improvement on search performance with less memory compared against the state of art. However, its performance relies heavily on the size of the codebook which is used to generate VLAD representation. It indicates better accuracy needs higher dimensional representation. Thus, more memory overhead is needed. In this paper, we enhance VLAD image representation by using two level hierarchical-codebooks. It can provide more accurate search performance while keeping the VLAD size unchanged. In addition, hierarchical-codebooks are used to construct multiple inverted files for more accurate non-exhaustive search. Experimental results show that our method can make significant improvement on both VLAD image representation and non-exhaustive search.

***Keywords:*** hierarchical-codebook; enhanced VLAD; projected residual vector quantization; multiple inverted files

**http://dx.doi.org/10.3837/tiis.2016.07.022** **ISSN : 1976-7277**

## 1. Introduction

Content-based image retrieval (CBIR) is a historical line of research in Multimedia and it has received extensive attentions [1-12]. The main task of it is image similarity search, i.e., finding the images in a database that are similar to a query image. There are two problems that all practical system must be solved, which are search time and storage size. To achieve desirable performance, common solutions use a set of local descriptors which is extracted from images called "bag of descriptors". However, the prohibitive storage cost restricts the use of bag of descriptors because of the reason that the size of it is usually larger than the image itself. Many methods resort to compute a lightweight signature using the bag of descriptors to represent images. The most famous representative one is the bag-of-words representation (BOF) [13] which is borrowed from text retrieval. BOF aggregates a bag of descriptors into a single vector which is obtained as the histogram of the assignment of all image descriptors to code words. Generally, to achieve comparable search performance, the number of code words should be large enough, e.g., up to millions. Therefore, BOF can restrict some really application because of the prohibitive memory overhead.

In recent years, another image index scheme was provided [14] [15], i.e., Vector of Locally Aggregated Descriptors (VLAD) which is a simplified version of Fisher Vector (FV) [16] [17]. It models the local descriptors space by a small codebook obtained by clustering a large set of local descriptors. The model is simply the sum of residual vectors, i.e., all centered descriptors from their nearest code words. Each code word is responsible for one subvector. The final signature is obtained by a concatenation of all subvectors. Comparing with BOF, VLAD can obtain a comparable search quality with less memory. Recently, several variations have been proposed to improve the quality of VLAD representation [18].

In this paper, we propose to leverage two level hierarchical-codebooks to enhance the VLAD representation. The key idea of this paper is to generate smaller residual while keeping the size of VLAD unchanged. The hierarchical-codebook is also applied to generate multiple inverted files to improve non-exhaustive search.

This paper is organized as follows. Section 2 briefly reviewed VLAD formulation and some improvements, while section 3 describes our enhanced VLAD, encode strategy and the non-exhaustive search strategies based on multiple inverted files. Experimental results validate our algorithm in section 4. The conclusion is obtained in section 5.

## 2. Related Work

VLAD [14] [15] is an image indexing technique that produces a vector representation $\mathcal{V}$ from a set $X = \{x_1, \cdots, x_N\}$ of N local d-dimensional descriptors extracted from a given image. Similar to BOF, a visual codebook $C = \{c_1, \cdots, c_K\}$ is learnt off-line. The codebook is formally used as a quantization function assigning any input local descriptor to its closest centroid (code word) as

$$Q: R^d \rightarrow C \subset R^d \tag{1}$$

$$x \longmapsto Q(x) = \arg\min_{c_i \in C} \|x - c_i\| \tag{2}$$

Where $\|\cdot\|$ refers to L2 norm.

Then aggregation operation is performed as follows. For each quantization index $i \in [1 \cdots K]$, a d-dimensional vector $v^i$ is obtained by accumulating the residual vectors which are generated by subtracting descriptors from their least distance code word as equation 3.

$$v^i = \sum_{x:Q(x)=c_i} x - c_i \tag{3}$$

The VLAD representation can be obtained by concatenating all K d-dimensional vector into a D=K×d dimensional vector as $v = \left[(v^1)^T \cdots \left(v^K\right)^T\right]^T$. An example of VLAD generation process is illustrated as **Fig. 1**.

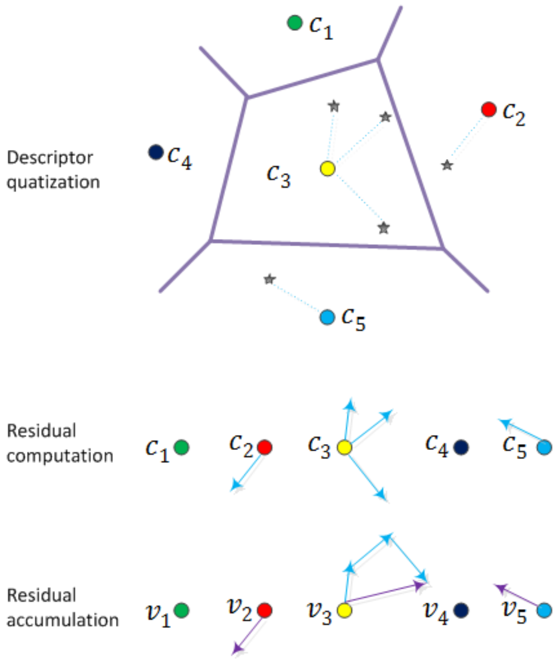



**Fig. 1.** An illustration of VLAD generation process

Two normalizations are then implemented to suppress local burst appearance of an image. Firstly, a component-wise nonlinear operation is applied: each component $v_j (1 \le j \le D)$ is modified as $v_j = \text{sign}(v_j) \times \|v_j\|^{\alpha}$, where the quantity α is a parameter such that $\alpha \le 1$. This is the so-called "power-law normalization" [9], which is motived in [7] by the presence of burst in natural image. Finally, VLAD vector is L2-normalized as $v = v/\|v\|$.

To satisfy memory constraint, VLAD memory footprint can be reduced significantly by performing a jointed optimized succession of dimension reduction and compression [14][15] with Product Quantization (PQ) [19]. Firstly the dimensionality of VLAD is reduced to D'<D components by PCA. Subsequently, after a random rotation that balances the subvectors of reduced vector, PQ [19] splits it into M subvectors, which are separately quantized with a k-means quantizer. This compression scheme allows the computation of distance between a query and a set of vectors in compressed domain. It does not require the explicit decompression of the database vectors and is therefore very fast. The choice of D' and M is tuned thanks to an optimization procedure that solely relies on a reconstruction criterion.

The original normalizations are prone to putting much weight on burst visual features, resulting in a sub optional measure of image similarity. To alleviate the problem, R.

Arandjelovic et.al [18] proposed a new normalization, termed intra-normalization (IN). The key idea of it is for each code word to L2 normalize the residuals summation within each VLAD block as

$$v^i = v^i / \|v^i\|, 1 \le i \le K \tag{4}$$

Comparing against the power-law normalization which discounts the burst effect, the inter-normalization fully suppresses the burst effect [18] [20].

# 3. Enhanced VLAD

## 3.1 EVLAD Framework

In this section, we introduce a technique, called Enhanced VLAD (EVLAD), to improve VLAD representation by utilizing two level hierarchical-codebooks.

For VLAD representation, our observation is that VLAD representation becomes more powerful by increasing the size of codebook. The main reason can be that the smaller quantization errors are generated by using the bigger codebook. However, with the original VLAD framework, the higher dimensional VLAD representation will be generated with the bigger codebook and increase the memory overhead inevitably. This inspires us to partition the local descriptors spaces with finer-granularity and to generate the smaller quantization errors while keeping the VLAD vector dimensionality unchanged. It is proposed to solve question by using two level hierarchical-codebooks and the resulting VLAD is termed the Enhanced VLAD (EVLAD).

Our two level hierarchical-codebooks learnt off-line are something like as hierarchical k-means. However, ours admits that the branches of each level can be different. For the sake of clarity, some notions are described as follows. The first level codebook with K centroids (code words), which is denoted by $\mathrm{C}^1 = \{\mathrm{c}_1, \cdots, \mathrm{c}_\mathrm{K}\}$, partitions the N training local descriptors into K different subset actually. The local descriptors in the same subset are further used to train L centroids which are served as the second level codebooks and denoted by $\mathrm{C}^2 = \left\{\{\mathrm{c}_1^1, \cdots, \mathrm{c}_\mathrm{L}^2\}, \cdots, \{\mathrm{c}_1^\mathrm{K}, \cdots, \mathrm{c}_\mathrm{L}^\mathrm{K}\}\right\} = \{\mathrm{C}_1^2, \cdots, \mathrm{C}_\mathrm{K}^2\}$. Here, the parameters K and L are maybe different. The two level codebooks are formally used as 1+K quantization functions assigning any input local descriptors into its two level closest centroid separately as

$$Q_1 = R^d \to C^1 \subset R^d \tag{5}$$

$$\mathrm{x} \mapsto Q_1(x) = \arg\min_{c_k \in C_1} \|x - c_k\| \tag{6}$$

$$Q_2^i = R^d \to C_i^2 \subset R^d, 1 \le i \le K \tag{7}$$

$$x \mapsto Q_2^i(x) = \arg\min_{c_k^i \in C_i^2} \|x - c_k^i\|, 1 \le i \le K \tag{8}$$

Where $\|\cdot\|$ refers to the $L_2$ norm.

As far as aggregating pipeline is concerned, EVLAD is different from the original VLAD. Let the first level codebook size K, EVLAD vector as VLAD vector also has K subvectors and code word is responsible for a subvector which accumulates the corresponding residual vectors. However, the generating residual vector bases are different. For a local descriptor x, EVLAD handle it as follows. As equation 9, the first level codebook is used as a quantizer and the quantization index i of the descriptor x determine the related subvector to which x's residual vector should add.

$$i = \mathrm{argmin}_{k: c_k \in C_1} \|x - c_k\| \tag{9}$$

But the exact residual vector about descriptor x is generated based on the second level code word $c_j^i$, where $c_j^i$ in $C_i^2$ has the least distance to x. And the residual vector will be added to the $i^{th}$ sub-vector of EVLAD vector as equation 10.

$$v^i = \sum_{x:Q_1(x)=c_i \& Q_2^i(x)=c_j^i} x - c_j^i \tag{10}$$

An example of EVLAD descriptor generation process is illustrated as **Fig. 2.**

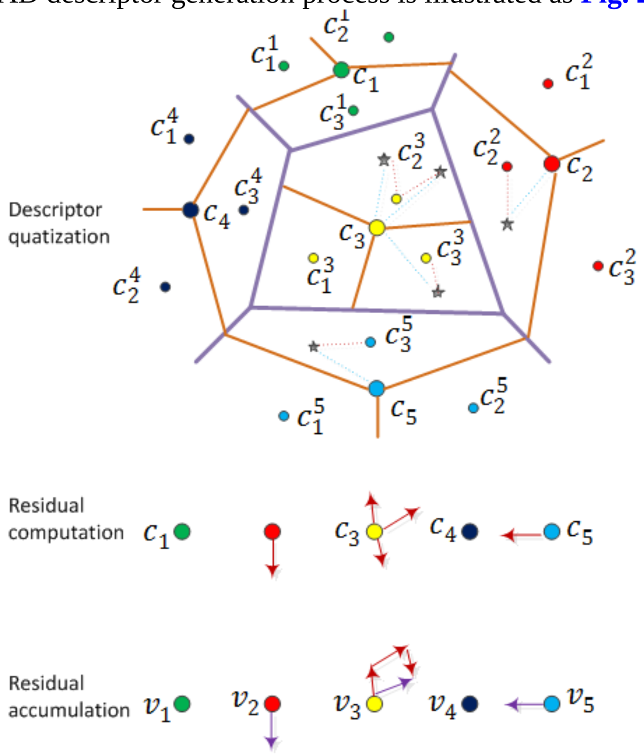



**Fig. 2.** An illustration of EVLAD descriptor generation process

In this example, the descriptor space is first partitioned into 5 quantization cells, and the cell centers, i.e. the first level code words, are represented in different color circles. Further, each cell is split into 3 new parts, and each center, i.e. the second code word, is represented as same color smaller circle. Residual computation is obtained by subtracting descriptor from the nearest second level code word. Thus, the accumulated residuals represented as purple arrow are smaller than residuals generated with VLAD framework shown as **Fig. 1.**

Our EVLAD is then obtained by two normalization operations. First, a subvector-wise L2-normalization is applied as $v^i = v^i/\|v^i\|$ which is termed intra-normalization [18] followed by concatenating all K d-dimensional subvectors into a D = K×d dimensional vector. The power-law normalization applied in original VLAD formulation is omitted in EVLAD formulation. Finally, EVLAD vector is $L_2$-normalized as $v = v/\|v\|$. In summary, our EVLAD is described as **Algorithm 1**.

**Algorithm 1** Computation of EVLAD descriptor $V$ from a set of descriptors $X = \{x_1, \cdots, x_N\}$. The hierarchical codebooks $C^1 = \{c_1, \cdots, c_k\}$ and $C^2 = \{C_1^2, \cdots, C_K^2\}$ are learnt on a training set by using k-means, where $C_k^2 = \{c_1^k, \cdots, c_L^k\}, k = 1 \cdots K$.

*for* $k$=1:$K$

$v^k = 0_d$ //initialization

*for* $n$=1:$N$

$i = \arg\min_{k:c_k \in C^1} \|x_n - c_k\|$

$j = \arg\min_{l:c_l^i \in C^2} \|x_n - c_l^i\|$

$v^i = v^i + x_n - c_j^i$ // aggregate residual vectors

*for* $k$=1:$K$

$v^i = v^i / \|v^i\|$ // intra-normalizatio

$v = [(v^1)^T \cdots (v^K)^T]^T$

$v = v / \|v\|$ // $L_2$-normalization

### 3.2 Complexity Analysis

The complexity of our EVLAD is described as follows. To store the hierarchical-codebook, the storage complexity is $O(KLd)$. Because the parameters K and L are usually small, the memory overhead of the hierarchical-codebook can be omitted.

Given an image with N local descriptor, to generate the EVLAD representation, the computational complexity is $O(N(K+L)d)$. Compared against the original computational complexity $O(NKd)$, our algorithm increases the computation complexity moderately.

### 3.3 From vector to code

The resulting EVLAD vectors generated as **Algorithm 1** are high dimensional and the required storage overhand is prohibitive for large scale database. In this section, we address the general problem of encoding a high dimensional image descriptor with only several bytes while keeping desirable discrimination. Given an image descriptor, a D-dimensional input vector, we want to generate a B bits code to encode the image representation, such that the approximate nearest neighbors of a (non-encoded) query vector can be searched efficiently in a set of encoded database vectors. This problem is handled by projected residual vector quantization [23] (PRVQ) which can satisfy memory constraint and offer desirable search accuracy.

PRVQ quantizer consists of multiple stage-quanitzers. In each stage, a projection matrix is used to project the original vector into a low dimensional space. The projected vectors are used to learn a stage-quantizer by clustering algorithm such as k-means. Then, the quantization errors of the projected vectors are reprojected by the transposition of the projection matrix into the original vector space to get new residual vectors set which are further used to learn a new stage projection matrix and stage-quantizer. We repeat this process m times, a quantizer composed of m stage-quantizers can be learnt. When quantizing a database vector, the projection matrixes and quantizers constructed in each training stage are used to generate a tuple of indices which correspond to the serial number whose cluster centroid has the least distance to the vector's stage residual. Let each individual stage-quantizer have ks reproduction values. In general, to limit the assignment complexity, ks is small (e.g., ks = 256). However, the set K of centroids induced by PRVQ quantizer is large: $K = (ks)^m$. When a query q is submitted, the distance between q and a database vector x is used as similarity measure. Comparing with symmetric distance computation (SDC), the asymmetric distance computation (ADC) with the smaller distance error is computed as

$$\|q-x\|^2 = \left\|q-\sum_{i=1}^{m}\left((p^i)^T \cdot c_j^i\right)\right\|^2 = \|q\|^2 + \left\|\sum_{i=1}^{m}\left((p^i)^T \cdot c_j^i\right)\right\|^2 - 2\sum_{i=1}^{m}\langle p^i \cdot q, c_j^i\rangle \quad (13)$$

Where $c_j^i$ and $p^i$ are the jth centroid and the projection matrix respectively in stage i. The term $\|q\|^2$ affects equally for all database vectors and can be neglected when sorting the offline. The summation term $\sum_{i=1}^{m}\langle p^i \cdot q, c_j^i\rangle$ can be read from look-table. So the search result can be obtained quickly. The detailed introduction of PRVQ see reference [23].

### 3.4 Non-exhaustive search

PRVQ provides an approximate solution for fast nearest neighbor search and reduces the memory constraints significantly for storing the EVLAD descriptors by encoding them into compact code. However, the search is exhaustive. In this section, we introduce a popular non-exhaustive solution by use of multiple inverted files. It has been found that multiple inverted files, obtained by multiple independent quantizers (codebooks), are able to achieve practically good recall and speed. In this section, a simple but effective algorithm is introduced to generate multiple quantizers by use of the two level hierarchical-codebooks.

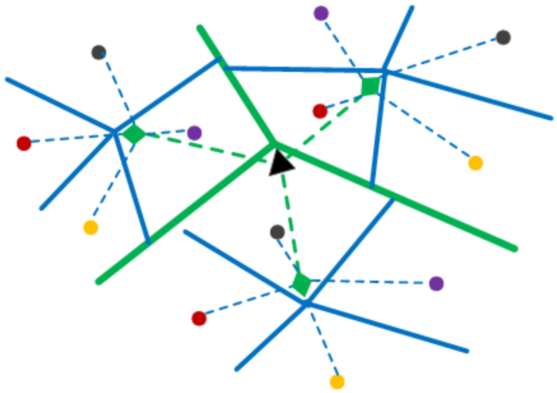


**Fig. 3.** An illustration of multiple quantizers generated from the second code words represented as color circles and the same color code words are grouped into the same quantizer.

The multiple quantizaters devote to group the second level code words into L groups and each group with K code words. An example is shown as **Fig. 3**. In this example, the second code words are grouped into 4 groups which are illustrated as 4 different color circles and each group has 3 code words. How the multiple quantizers are generated effectively? A simple but suboptimal scheme can be adopted as follows. First K code words are selected out randomly without replacement from each of the second level *K* codebooks to form one quantizer. All *L* quantizers are constructed by repeating the above process *L* times. We refer to these *L* quantizers as *random-multiple-quantizers* (RMQs). However, a bad case can be present, in which, the code words in some quantizers are too concentrated while those in other quantizers are too loose. It can produce large quantization error when they are used to create multiple inverted files for large scale retrieval thus deteriorates the search performance. A better strategy should disperse the code words in every quantizer so as to minimize the quantization errors.

**Algorithm 2** constructs *L* quantizers by dispersing the second level *K* codebooks incrementally into *L* groups and each group (can be seen as a new codebook, i.e., quantizer) is with $K$ code words. First, we use $C_1^2$ to initialize $L$ groups by assigning $L$ code words into $L$ groups without replacement. The remainder $K$-1 codebooks are progressively dispersed into $L$ groups. When dealing with the codebook $C_k^2, 2 \leq \mathrm{k} \leq \mathrm{K}$, we assign all its $L$ code words into $L$ different groups without replacement. **Algorithm 3** shows us how to disperse the code words of $C_k^2$. It is based on the distances matrix $D_{L \times L}$which stores the distances between centroids of all the groups and codebook $C_k^2$. In **Algorithm 3**, *idx* is an indicator and $idx(j)$ indicates the group number to which the $j^{th}$ code word of codebook $C_k^2$ should be assigned. Firstly, an initialization is implemented. Then, a rectification whose objective is maximizing the sum of distances between the group centroids and the code words of codebook $C_k^2$ is performed. After the indicator *idx* is obtained by **Algorithm 3**, it is used to aid to assign code word of codebook $C_k^2$ into corresponding group. We then update the centroid of group by recalculating the mean of all code words of each group. We repeat the above process $K$-1 times until $K$-1 codebooks are handled out. The final $L$ groups (quantizers) are termed as *enhanced-multiple-quantizers* (EMQs).After the $L$ EMQs are built, they can be used to generate $L$ inverted files when the database vectors are quantized.

**Algorithm 2** *Computing L quantizers Q from two level codebooks* $C^1 = \{c_1, \cdots, c_k\}$ and $C^2 = \{C_1^2, \cdots, C_K^2\}$, *where* $C_k^2 = \{c_1^k, \cdots, c_L^k\}, k = 1 \cdots K$.

*for l*=1:*L*
  $Q^l = \{c_l^1\}$ // *initialization by using* $C_1^2$
  $M^l = mean(Q^l)$ // $M^l$ *is the centroid of quantizer* $Q^l$
*for k*=2:*K*
  $D(i,j) = d\left(M^i, c_j^k\right)$ // *compute the initial distance*
  *idx* = *compute_indicator*(*D*) // *idx*(*j*), *an indicator computed as Algorithm 3,*
  *for j*=1:*L*
    $Q^{idx(j)} = Q^{idx(j)} \cup c_j^k$ // *update quantizers*
    $M^{idx(j)} = mean\left(Q^{idx(j)}\right)$ // *update quantizer centroids*

**Algorithm 3** *Computing idx from distance matrix D*

*for l* = 1:*L*
  *idx*(*l*) = *l* //*initialize indicator*
*for i* = 1:*L*-1
  *for j* = *i* +1:*L*
    *if* $D(idx(i), i) + D(idx(j), j) < D(idx(j), i) + D(idx(i), j)$
      *swap*(*idx*(*i*), *idx*(*j*)) // *swap indicator*

## 4. Experimental Results and Analysis

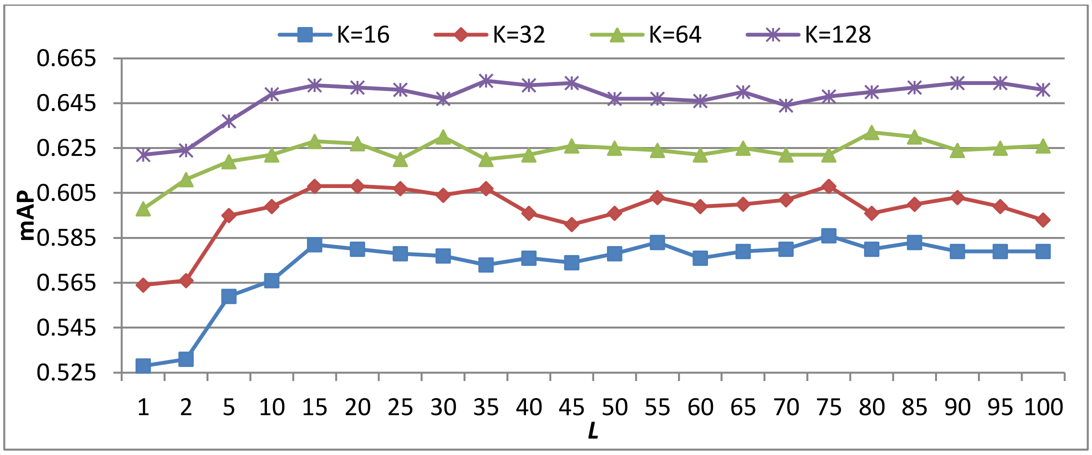
In order to evaluate our work, we adopt some public datasets and corresponding evaluation protocols that are usually considered in this context.

### 4.1 Dataset and Evaluation Protocol

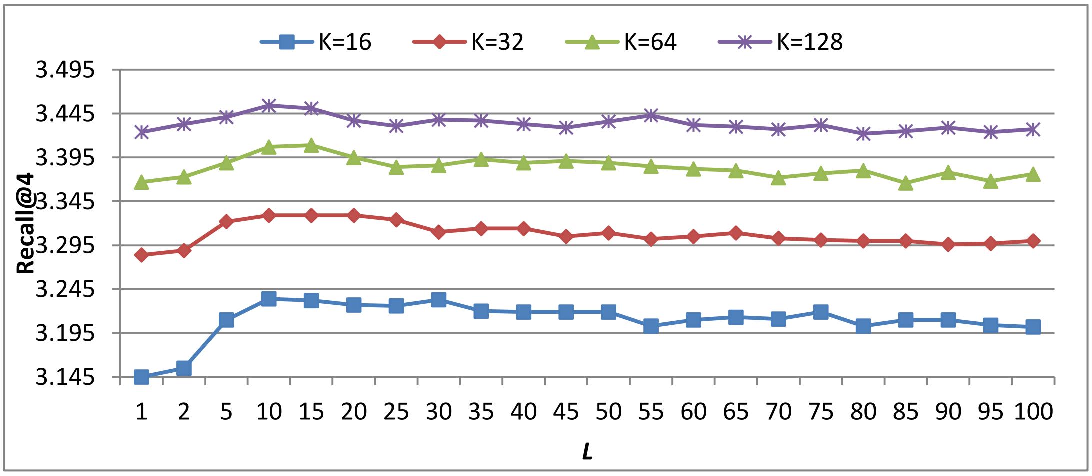
**Holidays dataset.** Holidays dataset [21] is a collection of 1491 high resolution personal photos of different locations and objects, 500 of them being used as queries, with the query removed from the ranked list. The accuracy is measured by the mean Average Precision(mAP).
**UKB dataset.** The UKB dataset [22] contains images of 2550 objects, each of which is represented by 4 images. The most common evaluation metric for this dataset counts the average number of relevant images (including the query itself) that are ranked in the first four positions. This corresponds to 4 times the recall@4 measure, i.e., the best performance is 4.
**Holidays_Flickr1M dataset.** Large scale evaluation is performed on Holidays merged with Flickr1M which consists of 1 million images collected from Flickr. The accuracy is measured by mAP.

### 4.2 Performance of EVLAD

### (a) Impact of the codebook size




(a)




(b)

**Fig. 4.** An illustration of the impact of the parameters K and L for EVLAD. (a) Holidays, (b) UKB

We first test the impact of different K and L on search accuracy with the hierarchical codebook learnt with the independent training dataset Flickr60k. The results are shown in **Fig. 4**. When L equals to 1, our EVLAD is equivalent to $VLAD_{IN}$ [18]. From **Fig. 4**, we can see that our EVLAD outperform the $VLAD_{IN}$ formulation with both the Holidays dataset (**Fig. 4(a)**) and UKB dataset (**Fig. 4(b)**). Fixing the parameter L, the EVLAD performance gets better by increasing the value of the parameter K. It indicates that higher dimensional representation usually provide better accuracy. As we can see from **Fig. 4(a)** that EVLAD improves obviously before reaching its performance peak (L=15) by increasing the value of the parameter L for different configurations of the parameter K. In the cases of L>15, EVLAD does not improve obviously. On the contrary, it starts to drop slowly or fluctuate smoothly. The same situation happens to the UKB dataset (**Fig. 4(b)**), and EVLAD reaches its performance extreme point in the case of L = 10. In the following comparison against the state of art, we will still just consider the case of K = 64 and L = 15 which we still term it EVLAD.

### (b) Comparison with state of the art

**Table 1.** Comparison of EVLAD with state-of-the-art on Holidays

| Descriptor | *K* | *D* | Raw descriptors | After dimensionality reduction to 128 compoments | Encoded into 8 bytes with PRVQ (P=32) |
|---|---|---|---|---|---|
| BOW[14] | 20k | 20k | 0.404 | 0.444 | --- |
| Fisher[14] | 64 | 8192 | 0.495 | 0.492 | --- |
| VLAD | 64 | 8192 | 0.561 | 0.576 | 0.575 |
| $VLAD_{IN}$ | | | 0.594 | 0.611 | 0.584 |
| EVLAD | 64 | 8192 | **0.635** | **0.637** | **0.606** |

**Table 1** compares our EVLAD representation with the result of literature [14] (mAP performance for Holidays), which includes different vector representations, in particular, FV and BOF baseline. For VLAD and $VLAD_{IN}$ we use the code provided by the authors and perform it by ourselves. With the size of 8192, our EVLAD outperforms all the baselines. Particularly, our EVLAD outperforms BOF by 57.1% and Fisher by 28.2% and the original VLAD by 13.3%. Although comparing with the best baseline $VLAD_{IN}$, the mAP increasement is about 6.9%, from 0.594 to 0.635. Column 4 and column 5 of **Table 1** show the relative improvement of EVLAD when applying dimensionality reduction and using compact codes obtained by PRVQ [23].

**Table 2.** Comparison of EVLAD with state-of-the-art on UKB (Recall@4 performance)

| Descriptor | *K* | *D* | Raw descriptors | After dimensionality reduction to128 compoments | Encoded into 8 bytes with PRVQ (P=32) |
|---|---|---|---|---|---|
| BOW[14] | 20k | 20k | 2.87 | 2.95 | --- |
| Fisher[14] | 64 | 8192 | 3.09 | 3.09 | --- |

| | | | | | |
|---|---|---|---|---|---|
| VLAD | 64 | 8192 | 3.20 | 3.19 | 3.19 |
| $VLAD_{IN}$ | | | 3.36 | 3.32 | 3.32 |
| EVLAD | 64 | 8192 | **3.51** | **3.44** | **3.44** |

Similarly, with UKB dataset, to validate our approach, we carry out the same performance evaluation and the results are shown in **Table 2**. From **Table 2**, we also can see that our EVLAD shows the same outstanding performance. Particularly, our EVLAD outperforms BOF by 22.3% and Fisher by 13.6% and the original VLAD by 9.7%. Although comparing with the best baseline $VLAD_{IN}$, the Recall@4 increasement is about 4.5%, from 3.36 to 3.51. Similarily, Column 4 and 5 of **Table 2** show the relative improvement of EVLAD when applying dimensionality reduction and using compact codes obtained by PRVQ [23].

### (c) Large scale image retrieval

**Table 3.** Large scale search on Holidays_Flickr1M (mAP performance)

| Descriptor | *K* | *D* | Raw descriptors | After dimensionality reduction to128 compoments | Encoded into 8 bytes with PRVQ (P=32) |
|---|---|---|---|---|---|
| VLAD | 64 | 8192 | 0.545 | 0.335 | 0.322 |
| $VLAD_{IN}$ | | | 0.576 | 0.358 | 0.328 |
| EVLAD | | | **0.607** | **0.374** | **0.339** |

With dataset up to 1 million images and compact image descriptor (128D), we test our EVLAD on large scale image retrieval by performing exhaustive nearest neighbor search with raw descriptors and compressed version generated with PCA projection. We also further encode the compressed vectors into compact codes with PRVQ[23] to satisfy the memory constraints. In addition, to validate the superiority of our EVLAD, we also perform VLAD and $VLAD_{IN}$ with the same setup as EVLAD. The test results are shown in **Table 3**. As we can see from **Table 3**, EVLAD show excellent performance in all different configurations.

### 4.3 Performance of non-exhaustive search

**Table 4.** non-exhaustive search combined with PRVQ (P=32) on Holidays+Flickr1M

| *K* | *L* | exhaustive | | SIF | | RMQs | | EMQs | |
|---|---|---|---|---|---|---|---|---|---|
| | | time | mAP | time | mAP | time | mAP | time | mAP |
| 32 | 8 | 1001491 | 0.339 | 29921 | 0.269 | 592257 | 0.337 | 584833 | **0.338** |
| 64 | 8 | 1001491 | 0.339 | 14541 | 0.276 | 376941 | 0.334 | 351773 | **0.337** |
| 128 | 8 | 1001491 | 0.339 | 6794 | 0.273 | 198273 | 0.314 | 152855 | **0.337** |
| 32 | 16 | 1001491 | 0.339 | 29755 | 0.284 | 891807 | 0.337 | 861585 | **0.339** |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 64 | 16 | 1001491 | 0.339 | 14612 | 0.283 | 622050 | 0.338 | 608996 | **0.339** |
| 128 | 16 | 1001491 | 0.339 | 7126 | 0.294 | 402635 | 0.332 | 386175 | **0.338** |

With the Holidays_Flickr1M dataset described with EVLAD, We evaluate our non-exhaustive search strategy combined with PRVQ. We perform three different non-exhaustive search strategies which are single inverted file (SIF), RMQs and EMQs respectively. For fair comparison, SIF traverses L inverted lists for a given query. The results in different settings are shown in **Table 4** and the time is measured by the average traversed vectors. As we can see from **Table 4** that SIF retrieval performance is the worst although it spends the least search time. Compared against exhaustive search, EMQs almost obtains the same performance while spending far less search time. RMQs obtain a little worse retrieval accuracy and spend a little more search time than EMQs. In conclusion, EMQs show the outstanding performance in search time and retrieval accuracy.

## 5. Conclusion

This paper has analyzed the VLAD representation, and shows that it leads to suboptimal results due to the limitation: to obtain better accuracy, VLAD should be with higher dimensional representation. We propose to leverage hierarchical codebooks to solve the problem. Our approach significantly outperforms the state of the art in term of search quality. In addition, 2-level hierarchical-codebooks are used to build multiple inverted files for more accurate and faster search. Experimental results show that our method can make significant improvement on both VLAD image representation and non-exhaustive search.

## Acknowledgement

This research is supported by the Special Fund for Earthquake Research in the Public Interest No.201508025, National Natural Science Foundation of China (NSFC) under Grant No.61272202.

## References

[1] T. Guan, Y.F. He, J. Gao, J.Z. Yang, J.Q. Yu, "On-Device Mobile Visual Location Recognition by Integrating Vision and Inertial Sensors," *IEEE Transactions on Multimedia*, vo. 15, no. 7, pp. 1688-1699, Nov. 2013. Article(CrossRef Link)

[2] R. Ji, X. Xie, H. Yao, W. Ma, "Mining City Landmarks from Blogs by Graph Modeling," *ACM Multimedia*, pp. 105-114, 2009. Article(CrossRef Link)

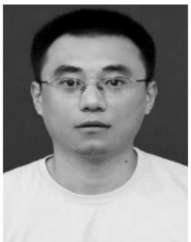
[3] Y. Gao, M. Wang, D. Tao, R. Ji, Q. Dai, "3-D Object Retrieval and Recognition with Hypergraph Analysis," *IEEE Transaction on Image Processing*, vol.21, no.9, pp. 4290-4303, Sept. 2012. Article(CrossRef Link)

[4] C. S. Anan and R. Hartley, "Optimized KD-trees for fast image descriptor matching," in *Proc. of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.1-8, June 23-28, 2008. Article(CrossRef Link)

[5] Ji RR, Duan LY, Chen J, Yao HX, Huang TJ et al, "Learning Compact Visual Descriptor for Low Bit Rate Mobile Landmark Search," *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCA)*, vol. 34, no. 2 pp. 2456-2463, 2011. Article(CrossRef Link)

[6] Ji RR, Duan LY, Chen J, Yao HX, "Location Discriminative Vocabulary Coding for Mobile Landmark Search," *International Journal of Computer Vision (IJCV)*, vol. 96, no.3, pp. 290-314, February, 2012. Article(CrossRef Link)
[7] Ji RR, Yao HX, Liu WL, Sun XS, Tian Q, "Task Dependent Visual Codebook Compression," *IEEE Transactions on Image Processing*, vol. 21, no.4, pp. 2282-2293, April, 2012. Article(CrossRef Link)
[8] Ji RR, Duan LY, Chen J, Huang TJ, Gao W, "Mining Compact Bag-of- Patterns for Low Bit Rate Mobile Visual Search," *IEEE Transactions on Image Processing*, vol. 23, no. 7, pp. 3099-3133, July, 2014. Article(CrossRef Link)
[9] Ji RR, Duan LY, Yao HX, Xie LX, Rui Y, et al, "Learning to Distribute Vocabulary Indexing for Scalable Visual Search," *IEEE Transactions on Multimedia*, vol. 15, no. 1, pp. 153-166, Jan. 2013. Article(CrossRef Link)
[10] Gao Y, Wang M, Li XL, Wu XD, "Visual-Textual Joint Relevance Learning for Tag-Based Social Image Search," *IEEE Transactions on Image Processing*, vol. 22, no. 1, pp. 363-376, Jan. 2013. Article(CrossRef Link)
[11] Ji RR, Gao Y, Hong RC, Liu Q, Tao DC, et al., "Spectral-Spatial Constraint Hyperspectral Image Classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, no. 3, pp. 1811 -1824, March, 2014, Article(CrossRef Link)
[12] Guan T, He YF, Gao J, Yang JZ, Yu JQ, "On-Device Mobile Visual Location Recognition by Integrating Vision and Inertial Sensors," *IEEE Trans, Multimedia*, vol. 15, no. 7, pp. 1688 -1699, Nov., 2013. Article(CrossRef Link)
[13] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *Proc. of IEEE International Conference on Computer Vision (ICCV)*, pp. 1470-1477, Oct. 13-16, 2003. Article(CrossRef Link)
[14] H. Jegou, M. Douze, C. Schmid, and P. Perez, "Aggregating local descriptors into a compact image representation," in *Proc. of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3304-3311, June 13-18, 2010. Article(CrossRef Link)
[15] H. Jegou, F. Perronnin, M. Douze, J. Sanchez, P. Perez, and C. Schmid, "Aggregating local image descriptors into compact codes," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 34, no. 9, pp.1704–1714, Sept., 2012. Article(CrossRef Link)
[16] F. Perronnin and C. R. Dance, "Fisher kernels on visual vocabularies for image categorization," in *Proc. of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1-8, June 17-22, 2007. Article(CrossRef Link)
[17] F. Perronnin, J.Sanchez, and T. Mensink, "Improving the Fisher kernel for large-scale image classification," *European Conference on Computer Vision (ECCV)*, pp.143-1560, Sept. 5-11, 2010. Article(CrossRef Link)
[18] R. Arandjelovic and A. Zisserman, "All about VLAD," in *Proc. of IEEE International Conference on Computer Vision and Pattern Recognition(CVPR)*, pp. 1578-1585, June 23-28, 2013. Article(CrossRef Link)
[19] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 33, no. 1, pp. 117–128, Jan., 2011. Article(CrossRef Link)
[20] H. Jegou, M. Douze, and C. Schmid, "Improving bag-of-features for large scale image search," *International Journal of Computer Vision (IJCV)*, vol. 87, no. 3, pp.316–336, May, 2010. Article(CrossRef Link)
[21] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," in *Proc. of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.1-8, June 17-22, 2007. Article(CrossRef Link)
[22] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision (IJCV)*, vol. 60, no. 2, pp. 91–110, Nov., 2004. Article(CrossRef Link)
[23] B. Wei, T. Guan, J. Yu, "Projected residual vector quantization for ANN search," *IEEE multimedia*, vol. 21, no. 3, pp. 41-51,June-Sept., 2014. Article(CrossRef Link)

**Benchang Wei** received the Ph.D. degree from Huazhong University of Science and Technology, Wuhan, China in 2015. He is currently a lecturer in Hubei University of Automotive Technology. His research interests include mobile augmented reality and mobile visual search.



**Tao Guan** received the Ph.D. degree from the Huazhong University of Science and Technology, Wuhan, China in 2008. He is currently an associate professor in the School of Computer Science and Technology, Huazhong University of Science and Technology. His research interests include mobile visual search and mobile augmented reality.

**Yawei Luo** is currently a Ph.D. candidate at School of Computer Science & Technology of Huazhong University of Science & Technology, Wuhan, China. His research interests include augmented reality, 3D reconstruction and image processing.

**Liya Duan** received the Ph.D. degree from the Huazhong University of Science and Technology, Wuhan, China in 2011. She is currently a researcher in the Institute of Oceanographic Instrumentation at Shandong Academy of Sciences. Hers research interests include mobile visual search and mobile augmented reality.

**Junqing Yu** received the Ph.D. degree from Wuhan University, Wuhan, China in 2002. Currently, he is a professor in the School of Computer Science and Technology at Huazhong University of Science and Technology. His research interests include digital media processing and retrieval, multicore programming environment. He is a member of the IEEE and ACM.