Exclusive Visual Descriptor Quantization

Yu Zhang[1], Jianxin Wu[1], and Weiyao Lin[2]

[1] School of Computer Engineering, Nanyang Technological University, Singapore
[2] Department of Electronic Engineering, Shanghai Jiao Tong University, China

Abstract. Vector quantization (VQ) using exhaustive nearest neighbor (NN) search is the speed bottleneck in classic bag of visual words (BOV) models. Approximate NN (ANN) search methods still cost great time in VQ, since they check multiple regions in the search space to reduce VQ errors. In this paper, we propose ExVQ, an exclusive NN search method to speed up BOV models. Given a visual descriptor, a portion of search regions is excluded from the whole search space by a linear projection. We ensure that minimal VQ errors are introduced in the exclusion by learning an accurate classifier. Multiple exclusions are organized in a tree structure in ExVQ, whose VQ speed and VQ error rate can be reliably estimated. We show that ExVQ is much faster than state-of-the-art ANN methods in BOV models while maintaining almost the same classification accuracy. In addition, we empirically show that even with the VQ error rate as high as 30%, the classification accuracy of some ANN methods, including ExVQ, is similar to that of exhaustive search (which has zero VQ error). In some cases, ExVQ has even higher classification accuracy than the exhaustive search.

1 Introduction

Bag of visual words (BOV) has become a widely used practice in computer vision and related research, whose application include (but are not limited to) retrieval [1], scene recognition [2], and object categorization [3], etc. In spite of its simplicity, BOV achieves state-of-the-art performance in many problems.

In a classic BOV model, visual descriptors are first clustered to form a visual codebook, and the cluster centers are called visual codewords. A visual descriptor is then represented by the index of its nearest codeword. The procedure that maps a descriptor to its codeword index is called vector quantization (VQ). Finally, an image is represented by a histogram of the codeword indexes contained in this image. Classifiers are applied to these histograms to perform visual classification, e.g., in object or scene recognition. Recently, new BOV methods are also proposed, e.g., those based on the sparse coding [4][5].

In this paper, we limit our discussions to the classic BOV procedure, in which the vector quantization essentially performs a nearest neighbor (NN) search within all visual codewords, which is also the major speed bottleneck in applying BOV models. Two typical situations make this issue more prominent: large scale and real time. In large scale problems such as the TRECVID benchmark, weeks of time was needed even when computer clusters were used [6].

On the other hand, in real time applications where 30 or more images need to be processed per second, the VQ speed is again of paramount importance. We also observe that the VQ step occupies most of the processing time in BOV. Encoding an image in BOV is divided into three modules in [6]: point sampling, descriptor computation and VQ. Using the Caltech 101 [7] as an example, when we use densely sampled points and SIFT descriptors, the VQ time occupied from 87.53% to 96.56% of the total BOV processing time when the codebook size K varies from 256 to 1024.

Approximate nearest neighbor (ANN) search is a natural choice to speed up the VQ process. In this paper, we propose ExVQ, an efficient ANN method, which greatly accelerates the VQ speed, without affecting the classification accuracy of BOV models.

ANN methods, e.g., FLANN [8], have shown great speedup effects in searching within large point sets. It has already been used in vector quantization in [9]. In this paper, our goal is to further improve the VQ speedup by maintaining the classification accuracy. First, we systematically evaluate the effect of ANN VQ errors to the subsequent visual classification stage. We show that some ANN methods, even with up to 30% VQ error rates, do not reduce the classification accuracy in object and scene recognition tasks.

Next, we propose a new ANN method to speed up the VQ process. Existing ANN methods follow a branch and bound strategy: split the search space into regions according to certain criterion, and assume that a point and its nearest neighbor will fall into the same region. Thus, the algorithm only needs to examine the few points in one region. We call this strategy the *inclusive* strategy. The inclusive assumption, however, rarely holds in reality. Thus, searching only one region leads to high VQ error rate. More regions (e.g., nearby ones) must also be examined, which increases the cost of ANN search. For example, FLANN has a parameter that specifies how many regions are to be checked.

In ExVQ, we use an *exclusive* strategy instead. For a BOV model, all clusters of visual descriptors, not only the codewords, are divided into three regions, L, R, and I. Instead of insisting that nearest neighbors of visual descriptors from L must also reside inside L (i.e., inclusive), we only assume that such neighbors will not appear in R (and I is a buffer region which is ignored in this process). ExVQ thus do not search through codewords from clusters in R. We will show that our exclusive assumption can be ensured to be correct (or almost correct) by learning a classifier that assigns all clusters into appropriate regions L, R, and I. In reality, we perform several exclusions in ExVQ, organized in a hierarchical tree structure.

Overall, the following contributions are presented in this paper:

- A systematic evaluation of the effects of VQ errors caused by ANN. We reveal that a high VQ error rate (e.g., 30%) does not necessarily reduce the classification accuracy of the BOV model.
- An exclusion strategy for ANN search. We ensure the correctness of the exclusion by learning classifiers, which also gives a reliable estimation of the VQ error.

- ExVQ, an ANN system for vector quantization in BOV. ExVQ requires only about 10–20% of the VQ time of the classic VQ, or half of that of FLANN. And, all three methods have indistinguishable classification accuracies.

2 Related works

Vector quantization and its acceleration has long been studied in image and video coding. Tree-structure VQ methods such as TSVQ are popular because of its speed advantage [10]. Various branch-and-bound methods have also been used to find the exact or approximate NN with faster speed.

Tree structure is a useful tool to reduce search time. The hierarchical k-means tree can organize $O(n^m)$ items in a depth m tree, in which every internal tree node has n fan-out branches. A query then involves finding the path of the most similar nodes from the root of the tree toward a leaf node, which only requires nm distance computations. It has been successfully used in object retrieval from large object databases with tens of thousands of objects [11][12]. Randomized trees and extremely randomized clustering forests have also shown excellent speed and accuracy in matching features and detecting objects [13][14].

The branch-and-bound strategy, which usually utilizes linear projections to split the search space, is also proven valuable in ANN search. *kd*-tree splits the search space into halves using a hyperplane (which is a special case of linear projection) on the dimension with the largest variance and continues to split iteratively until each node contains only one point. Randomized *kd*-trees and ensembles of them have shown acceleration up to the 30x with approximately 90% correct nearest neighbors [8], comparing to the exhaustive nearest neighbor search of SIFT descriptors.

Spill tree [15] is similar to the proposed ExVQ, in that they both adopt overlap (buffer) between the two split search sets. Spill tree splits the search set using a random projection. The overlap contains those search points within a small distance to the projection boundary. The difference is, ExVQ learns an accurate SVM classifier from the available visual descriptors and uses it to determine: which clusters, rather than data points, form the search sets or the overlap. Visual descriptors in the two split search sets, especially in the overlap, have zero (or almost zero) NN search error, which is not ensured in the spill tree. Similar strategy using a buffer to eliminate the decision error is also used in multi-class problem [16][17]. However, the method in [17] is very slow, which may take days to learn on the same dataset used in this paper.

Another family of methods use multiple linear projections to map the feature vector (or visual descriptor) into a short string with few binary bits, e.g., locality sensitive hashing (LSH) [18], Hamming embedding [19], and spectral hashing [20]. This strategy is particularly attractive for processing huge datasets. Their storage requirements are greatly reduced, although their NN search accuracy may be significantly lower than ANN methods such as FLANN.

In recent years, sparse coding [4][5] is popular in object recognition and classification, such as Locality-constrained Linear Coding (LLC) [4]. Multiple

database points are assigned weights under the sparse constraint. In this paper, we will empirically compare ExVQ with LLC. Recently, product quantization [21] is also used for NN search. It can save search time and storage by quantizing the sub-vectors of the original high-dimension feature vector, which is, however, more suitable for retrieval applications rather than for visual recognition.

3 ExVQ: Exclusive Vector Quantization

In the BOV model, given a set of visual descriptors $X = \{\boldsymbol{x}_0, \boldsymbol{x}_1, ..., \boldsymbol{x}_{N-1}\}$, by applying the k-means clustering, the cluster centers are used as a codebook $\{\boldsymbol{c}_0, \boldsymbol{c}_1, ..., \boldsymbol{c}_{K-1}\}$, where $\boldsymbol{x}_i, \boldsymbol{c}_j \in \mathbb{R}^d, i = 0, 1, ..., N-1, j = 0, 1, ..., K-1$. A visual descriptor $\boldsymbol{x} \in \mathbb{R}^d$ is quantized by the index of its nearest codeword as:

$$nn(\boldsymbol{x}) = \underset{0 \le i \le K-1}{\arg\min}\ dist(\boldsymbol{x}, \boldsymbol{c}_i), \tag{1}$$

where $dist(\cdot, \cdot)$ can be any distance metric between two vectors. We consider the Euclidean distance in this paper, that is, $dist(\boldsymbol{x}, \boldsymbol{c}_i) = \|\boldsymbol{x} - \boldsymbol{c}_i\|_{\ell_2}$.

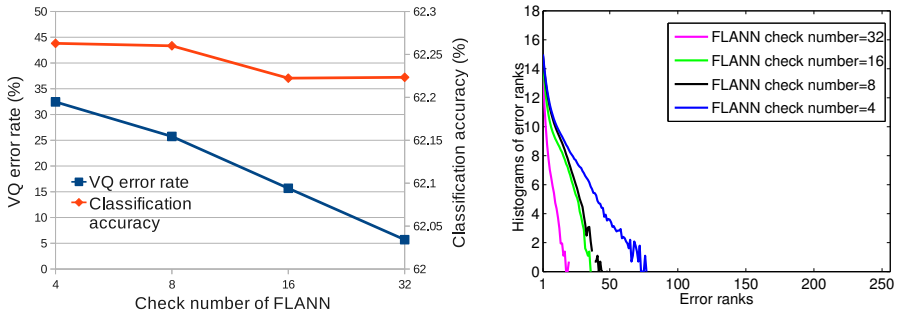
3.1 Effect of ANN VQ errors to classification

In an ANN method, we define the searched nearest neighbor index for a point $\boldsymbol{x}$ as $ann(\boldsymbol{x})$. During an ANN based VQ process, a VQ error happens when $ann(\boldsymbol{x})$ is different from $nn(\boldsymbol{x})$. Intuitively, a high VQ error rate will lead to more classification errors in the subsequent visual classification stage. However, we find empirically that this is not the case. When certain ANN methods are used, a high VQ error rate (e.g., 30%) does not necessarily hurt the classification accuracy. Thus, we have the freedom to allow some VQ errors in order to accelerate the VQ process.

Fig. 1a shows one example of how the classification accuracy varies with the VQ error rate of FLANN. Although VQ errors vary from 5.70% to 32.44%, the classification accuracies are almost the same. This result seems counter intuitive. We conjecture that *it can be explained by the properties of the VQ errors*: some VQ errors are benign and will have minimal effect on the classification performance.

To validate this conjecture, we first define the error rank of $ann(\boldsymbol{x})$ using the following procedure. We calculate the distance of $\boldsymbol{x}$ with each codeword $dist(\boldsymbol{c}_i, \boldsymbol{x}), i = 0, 1, ..., K-1$, and store them in an array in the ascending order. The error rank of $ann(\boldsymbol{x})$ is defined as the index of $dist(\boldsymbol{c}_{ann(\boldsymbol{x})}, \boldsymbol{x})$ in this sorted array. The error rank is 0 when $ann(\boldsymbol{x}) = nn(\boldsymbol{x})$. Then we define the benign VQ error: a small portion of VQs ($< 30\%$ or around)[1] has error and the error ranks are small. This definition provides us with an empirical way to evaluate the property of VQ errors. In Fig. 1b, we show the histograms of ANN error ranks using FLANN with different check numbers, where the vertical axis is shown

[1] We find 30% is valid for most cases. More results will be shown in the experiment.



(a) (b)

Fig. 1: (a) VQ error rate and classification accuracy (one round) using FLANN hierarchical k-means tree, on Caltech 101, when $K = 256$; (b) The histograms of ANN error ranks. The vertical axis is the frequency numbers shown *in natural logarithm*. We use FLANN hierarchical k-means tree with different check numbers on Caltech 101 when $K = 256$.

in natural logarithm. The error ranks of most VQ errors are close to 1, which supports our conjecture on benign VQ errors. More systematic results will be presented in Sec. 4.

3.2 ANN search: inclusion vs. exclusion

When inclusive methods (e.g., LSH and FLANN) search the nearest neighbor of a visual descriptor in the codebook $\{\boldsymbol{c}_0, \boldsymbol{c}_1, ..., \boldsymbol{c}_{K-1}\}$, they use linear projections to reduce the search range. With a linear projection $(\boldsymbol{w}, b), \boldsymbol{w} \in \mathbb{R}^d, b \in \mathbb{R}$, an inclusion strategy implicitly assumes that:

$$\forall\, \boldsymbol{x} \in \mathbb{R}^d, \mathrm{sgn}(\boldsymbol{w}^T \boldsymbol{c}_{nn(\boldsymbol{x})} + b) = \mathrm{sgn}(\boldsymbol{w}^T \boldsymbol{x} + b), \tag{2}$$

where $\mathrm{sgn}(\cdot)$ is the sign function.

However, in most cases, inclusive search methods fail to find the nearest neighbor in the reduced search range. Fig. 2 illustrates the inclusive NN search in the 2-D space. We use k-means to get 15 clusters from the randomly generated points. Each black star is the center of a cluster (i.e., codewords), which altogether form the codebook. Subject to the linear projection (the black line), 40% clusters (which are crossed by the line) violate the inclusive assumption. In reality, the inclusive assumption is violated even more seriously. Using a random linear projection on Caltech 101 ($K = 256$, $d = 128$), only about 10% clusters can guarantee Eq. 2. Thus the inclusive search methods usually need to check NN in multiple adjacent regions to maintain the search accuracy, which slows down the search speed.

We propose an exclusive NN search strategy to fulfill three goals: to reduce NN search time, to maintain a reasonable VQ error rate, and to ensure that the VQ errors are benign. In this strategy, the search range is reduced by excluding

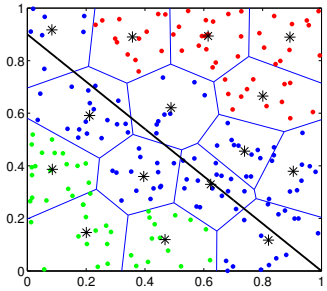


Fig. 2: Illustration of inclusive vs. exclusive NN search. The random projection violates the inclusive assumption. The exclusive strategy, however, can correctly excludes the red or the green clusters. This figure should be viewed in color.

a portion of clusters (rather than data points) from the whole search space subject to a linear projection. The exclusive NN search involves three index sets: the available search set C containing the indexes of all clusters to be searched, the positive and negative exclusion sets $C_{+1}(L)$ and $C_{-1}(R)$, which are the index sets of clusters that can be excluded.

We illustrate the idea of exclusive NN search in Fig. 2. We first assign the indexes of red and green clusters to C_{+1} and C_{-1}, respectively. Then, for a point above the linear projection, we search NN from the codewords within the red and blue clusters. Those green clusters (C_{-1}) can be safely excluded. In the inclusive strategy, for a point above the linear projection, its NN can be one of the blue-colored centers. Thus, regions in both sides of the black line are to be searched, that is, no cluster can be safely ignored.

In real-world NN search problems, a random linear projection may fail to produce high quality exclusion sets. Instead, we utilize existing examples to evaluate the irregular boundaries of exclusion clusters in C_{+1} and C_{-1}. We use visual descriptors in all exclusion clusters to train an accurate linear classifier such that a high quality linear projection direction can be learned. This linear projection can accurately split the exclusion clusters on two sides.

Specifically, first, we calculate the inner product of the codeword in each cluster $\boldsymbol{c}_i, i \in C = \{0, 1, ..., K-1\}$ with a random vector $\boldsymbol{r} \in \mathbb{R}^d$ and sort the values in the ascending order. Then, we choose the desired exclusion portion $0 < p < 1/2$ and assign the indexes of clusters which codewords are of the $p \cdot |C|$ largest (smallest) values of $\boldsymbol{r}^T \boldsymbol{c}_i$ to C_{+1} (C_{-1}). Next, we generate the positive training set $X_{+1} = \{\boldsymbol{x} | \boldsymbol{x} \in X, nn(\boldsymbol{x}) \in C_{+1}\}$ and the negative training set $X_{-1} = \{\boldsymbol{x} | \boldsymbol{x} \in X, nn(\boldsymbol{x}) \in C_{-1}\}$. Finally, we train a linear SVM classifier $(\boldsymbol{w}, b)$ using X_{+1} and X_{-1}:

$$\begin{aligned} \min_{\boldsymbol{w},b} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + \alpha \sum_{\boldsymbol{x} \in X_{+1}} \max(0, 1 - (\boldsymbol{w}^T \boldsymbol{x} + b))^2 \\ & + \alpha \sum_{\boldsymbol{x} \in X_{-1}} \max(0, 1 + (\boldsymbol{w}^T \boldsymbol{x} + b))^2, \end{aligned} \tag{3}$$

where α is the trade off parameter.

After the linear SVM classifier is trained, we obtain $(\boldsymbol{w}, b)$, C, C_{+1} and C_{-1}. For a visual descriptor $\boldsymbol{x} \in \mathbb{R}^d$, the exclusive NN search is defined as:

$$ann(\boldsymbol{x}) = \arg\min_i \|\boldsymbol{x} - \boldsymbol{c}_i\|_{\ell_2}, \tag{4}$$

$$\text{s.t. } i \in C - C_{-\text{sgn}(\boldsymbol{w}^T\boldsymbol{x}+b)}, \tag{5}$$

where $C - C_{-\text{sgn}(\boldsymbol{w}^T\boldsymbol{x}+b)}$ is the active search set of $\boldsymbol{x}$.

ExVQ and spill tree operates on different principles. ExVQ excludes one cluster entirely, which includes all points within the corresponding Voronoi cell. Thus, the nearest codewords of points in the rest search space never or rarely (depending on the training quality of the linear SVM classifier) exist in the exclusion set. In the spill tree, although a centroid may be far away from the exclusion region, its corresponding cluster (i.e., Voronoi cell) may have large overlap with it, that is, causing many VQ errors.

3.3 Tree-structured ExVQ

One single exclusion is not enough in real-world applications. In ExVQ, we use multiple linear classifiers with a hierarchical tree structure. We implement ExVQ in an L-level complete binary tree, where each node is indexed level by level from 0 to $2^L - 2$. The ith node is related to three cluster index sets: C^i, C^i_{+1} and C^i_{-1}, which are the available search set, and the positive and negative exclusion sets, respectively. First, in the root node, we set $C^0 = \{0, 1, ..., K-1\}$. Then, in the ith node, C^i_{+1} and C^i_{-1} are chosen from C^i by a random vector $\boldsymbol{r}_i \in \mathbb{R}^d$ as described in Sec. 3.2, and to train a linear SVM classifier $(\boldsymbol{w}_i, b_i)$ according to Eq. 3. Next, we compute the available search sets for the the left and right children of the ith node as $C^{2i+1} = C^i - C^i_{-1}$ and $C^{2i+2} = C^i - C^i_{+1}$, respectively. Finally, the whole tree can be built recursively from the root. This process is detailed in Algorithm 1.

After the tree is built, we get $\{(\boldsymbol{w}_i, b_i), C^i, C^i_{+1}, C^i_{-1}\}$ in every node. For a query $\boldsymbol{x} \in \mathbb{R}^d$, the NN search is processed as follows: first, starting from the root node, we compute $\boldsymbol{w}_i^T\boldsymbol{x} + b_i$. If the value is positive, the exclusion process goes to process its left child, otherwise to the right. This process continues iteratively until a leaf node is reached. Then we output the active search set of the leaf node. Finally, the NN of $\boldsymbol{x}$ is acquired by exhaustively searching the active search set. We fully specify ANN search by ExVQ in Algorithm 2.

In ExVQ, the tree level L and the exclusion portion p can influence the search speed and the VQ error rate. Applying ExVQ to a visual descriptor, the number of distance computations in the NN search is: $\hat{K} = L + K(1-p)^L$.

If we assume that the VQ error rates caused by the exclusions from the same level nodes in the tree are the same, the whole VQ error rate of ExVQ can be estimated by:

$$e_{vq} = 1 - \prod_{l=1}^{L}(1 - pe_l), \tag{6}$$

Algorithm 1 Build tree-structure ExVQ

Input: Descriptors $\{\boldsymbol{x}_0, \boldsymbol{x}_1, ..., \boldsymbol{x}_{N-1}\} \subseteq \mathbb{R}^d$ and K centers $\{\boldsymbol{c}_0, \boldsymbol{c}_1, ..., \boldsymbol{c}_{K-1}\}$, ExVQ tree level L and an exclusion portion parameter p.
$C^0 \leftarrow \{0, 1, ..., K-1\}$
for $id = 0$ **to** $2^L - 2$ **do**
  $C^{id}_{+1} \leftarrow \emptyset, C^{id}_{-1} \leftarrow \emptyset$, $parent \leftarrow 0$
  if $id > 0$ **then**
    $parent = \lfloor \frac{id-1}{2} \rfloor$
  end if
  if id is odd **then**
    $C^{id} \leftarrow C^{parent} - C^{parent}_{-1}$
  else
    $C^{id} \leftarrow C^{parent} - C^{parent}_{+1}$
  end if
  $a \leftarrow \emptyset$, $X_{+1} \leftarrow \emptyset, X_{-1} \leftarrow \emptyset$
  generate a random vector $\boldsymbol{r} \in \mathbb{R}^d$
  $a \leftarrow a \cup \boldsymbol{r}^T \boldsymbol{c}_i, \forall\ i \in C^{id}$
  for i in C^{id} **do**
    if $\boldsymbol{r}^T \boldsymbol{c}_i$ in the $p \cdot |C^{id}|$ biggest values of a **then**
      $C^{id}_{+1} \leftarrow C^{id}_{+1} \cup i$
      $X_{+1} \leftarrow X_{+1} \cup \boldsymbol{x}_j, \forall\ j, nn(\boldsymbol{x}_j) == i$
    end if
    if $\boldsymbol{r}^T \boldsymbol{c}_i$ in the $p \cdot |C^{id}|$ smallest values of a **then**
      $C^{id}_{-1} \leftarrow C^{id}_{-1} \cup i$
      $X_{-1} \leftarrow X_{-1} \cup \boldsymbol{x}_j, \forall\ j, nn(\boldsymbol{x}_j) == i$
    end if
  end for
  train a linear projection $(\boldsymbol{w}_{id}, b_{id})$ with positive set X_{+1} and negative set X_{-1}
end for
Output: $\{(\boldsymbol{w}_i, b_i), C^i, C^i_{+1}, C^i_{-1}\}$, $i = 0, 1, ..., 2^L - 2$.

where e_l is the VQ error rate caused by the exclusion in the l-th level. We note that e_l is the generalization error of the linear classifiers in level l. Take Caltech 101 as an example, when $K = 256$, $L = 10$, $p = 1/5$, we get $\hat{K}/K = 14.64\%$. If we use the maximum training error of all linear SVM classifiers as an estimation for e_l, we estimate the VQ error as $e_{vq} = 14.45\%$ using Eq. 6, which provides a reasonable estimation of the actual VQ error rate (which is 11.13%).

4 Experiments

We use 4 datasets in our experiments: Caltech 101 (15 + 20) [7], scene 15 (100 + rest) [3], sports 8 (70 + 60) [22], and indoor 67 (80 + 20) [23]. X+Y for each dataset means the number of training and testing images used in every category, respectively. Following the suggestion from [24], we use SIFT ($d = 128$) for Caltech 101, and the CENTRIST ($d = 256$) descriptor for other datasets. We use the LIBLINEAR package [25] to train a linear SVM classifier in each node

Algorithm 2 NN search using ExVQ

Input: A visual descriptor $\boldsymbol{q} \in \mathbb{R}^d$,
$\{(\boldsymbol{w}_i, b_i), C^i, C^i_{+1}, C^i_{-1}\}$, $i = 0, 1, ..., 2^L - 2$.
$id \leftarrow 0$
while $id < 2^L - 1$ **do**
  if $\boldsymbol{w}_{id}^T \boldsymbol{q} + b_{id} > 0$ **then**
    $id \leftarrow id \times 2 + 1$
  else
    $id \leftarrow id \times 2 + 2$
  end if
end while
Output:
$leaf = \lfloor \frac{id-1}{2} \rfloor$
if id is odd **then**
  $ann(\boldsymbol{q}) = \arg\min_{i \in (C^{leaf} - C^{leaf}_{-1})} \|\boldsymbol{q} - \boldsymbol{c}_i\|_{\ell_2}$.
else
  $ann(\boldsymbol{q}) = \arg\min_{i \in (C^{leaf} - C^{leaf}_{+1})} \|\boldsymbol{q} - \boldsymbol{c}_i\|_{\ell_2}$.
end if

of the ExVQ tree. The trade off parameter in Eq. 3 is $\alpha = 0.01$. After the BOV representation of all images are created, we use a one-vs-all SVM classifier with the χ^2 kernel $\kappa(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sum_{i=1}^{d} \frac{2x_{1i}x_{2i}}{x_{1i}+x_{2i}}$ [26][27] for classification. For each dataset, five training / testing sets are randomly split and average accuracy and VQ time of these five rounds are reported, if not otherwise specified.

4.1 VQ time vs. classification accuracy

First, we compare the VQ time and classification accuracy of ExVQ, FLANN and the exhaustive search, with different codebook size K. FLANN is a state-of-the-art ANN search method [8]. In all experiments, we use the hierarchical k-means tree (which leads to a higher classification accuracy than the randomized kd-trees) with default parameters in FLANN. For ExVQ, we fix the exclusion portion $p = 1/5$ and fix the tree level $L = 10$ for $K = 256$ and $L = 15$ for $K = 1024$. The results are shown in Table 1 ($K = 256$) and Table 2 ($K = 1024$).

ExVQ is much faster than FLANN while their classification accuracies are almost the same. In Table 1 and Table 2, the classification accuracy of ExVQ and FLANN on all four datasets have no statistically significant difference. However, ExVQ is about twice faster than FLANN no matter on the codebook with a small size ($K = 256$) or a large size ($K = 1024$). Both ExVQ and FLANN are much faster than the exhaustive search. ExVQ speed up NN search 5–10 times than the exhaustive search. Especially, in the indoor 67 dataset, when $K = 256$, ExVQ's classification accuracy is even significantly better than the exhaustive search.

When the codebook size is larger, the NN search speedup of ExVQ is more prominent. The search speedup of $K = 1024$ is about twice of that when $K =$

Table 1: VQ time T_i (seconds) and classification accuracy (%) of various methods. T_i/T_j is the speedup ratio between two methods. $i, j = 0, 1, 2$, $K = 256$.

	ExVQ				FLANN			Exhaustive	
Dataset	T_0	T_2/T_0	T_1/T_0	Accuracy	T_1	T_2/T_1	Accuracy	T_2	Accuracy
Caltech 101	129.2	4.86	2.83	61.87±0.25	365.0	1.72	62.22±0.72	627.8	62.63±0.55
Scene 15	358.6	5.77	2.22	81.16±0.45	797.6	2.60	81.17±0.14	2070.8	81.36±0.42
Sports 8	292.6	5.93	2.29	82.50±0.90	670.2	2.59	82.71±1.07	1736.2	82.33±1.28
Indoor 67	471.6	5.78	2.53	35.75±1.09	1193.4	2.28	35.04±1.28	2724.6	34.76±1.39

Table 2: VQ time T_i (seconds) and classification accuracy (%) of various methods. T_i/T_j is the speedup ratio between two methods. $i, j = 0, 1, 2$, $K = 1024$.

	ExVQ				FLANN			Exhaustive	
Dataset	T_0	T_2/T_0	T_1/T_0	Accuracy	T_1	T_2/T_1	Accuracy	T_2	Accuracy
Caltech 101	277.0	8.92	1.90	65.08±0.50	526.6	4.69	65.53±0.62	2472.0	65.79±0.53
Scene 15	660.2	12.36	2.41	83.17±0.32	1592.0	5.13	83.27±0.33	8162.4	83.53±0.20
Sports 8	556.4	12.39	2.92	83.42±0.77	1622.2	4.25	83.67±1.22	6892.4	84.17±1.11
Indoor 67	885.0	12.22	2.83	38.52±1.96	2503.2	4.32	38.91±1.30	10818.6	38.94±1.33

256, which is similar for FLANN. However, the larger the codebook size is, the more tree level ExVQ will need to achieve the speedup. This requirement usually costs more time in building the tree, mainly in training the linear SVM classifiers. When $K = 256$, building the 10-level tree costs 18.8–56.6 seconds on different datasets. The building time increases to 326–617 seconds using 15 levels when $K = 1024$. In the experiments, we keep the number of training descriptors from 40468 to 76219, to restrict the training time of linear SVM classifiers. Note that the classifiers are trained offline, and are trained for only once.

We also test a smaller check number for FLANN. When the check number is 4, FLANN is faster than default check number 32, but its VQ time is still at least 67% longer than ExVQ (or higher than 67% on different datasets). Furthermore, when $K = 1024$, the classification accuracy of FLANN with check number 4 is statistically significantly lower than that of ExVQ.

4.2 Properties of VQ errors

For these datasets, the VQ error rates of ExVQ and FLANN are shown in Table 3. In some cases, the VQ error rates of ExVQ and FLANN are close. In many cases, the VQ error rate of ExVQ is higher than that of FLANN. However, as observed from Table 1 and Table 2, the difference in VQ error rates does not have impact on the classification performance. In fact, if we increase L and reduce p, ExVQ can easily achieve both faster VQ speed and smaller VQ error rate than FLANN, with the cost of larger storage.

In practice, we don't need very large L, because ExVQ has benign VQ errors. We further evaluate the VQ errors generated by ExVQ in Fig. 3, which plots the histograms of error ranks for ExVQ. ExVQ error ranks are all within a small

Table 3: VQ error rates (%) of ExVQ and FLANN.

Dataset	$K = 256$		$K = 1024$	
	ExVQ	FLANN	ExVQ	FLANN
Caltech 101	11.13	4.91	14.15	19.32
Scene 15	4.99	3.20	18.64	7.14
Sports 8	5.82	4.23	19.19	6.63
Indoor 67	6.03	5.81	16.90	6.26

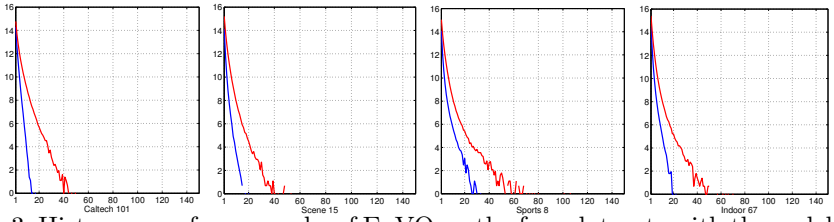


Fig. 3: Histograms of error ranks of ExVQ on the four datasets with the codebook size $K = 256$ (blue) and $K = 1024$ (red). The horizontal axis is the error rank and the vertical axis shows the frequency number plot in natural logarithm.

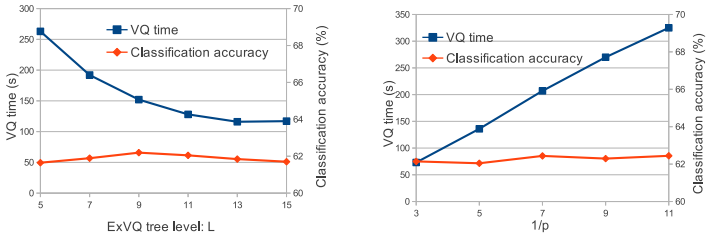
range (compared to the codebook size) close to 1, which we believe are benign to classification.

When the codebook size is larger, the maximum error rank is also lager. The maximum error rank for $K = 1024$ (red curve) is 2.57–3.15 times of that for $K = 256$ (blue curve). One may conjecture that the reduction in the number of training descriptors of each codeword makes the linear classifiers inaccurate when K is larger. However, when we increase the training descriptors for $K = 1024$, we find that the maximum error rank is almost the same as that in Fig. 3. In fact, the VQ error rate is mostly dependent on the tree level L and the exclusion portion p.

4.3 Effects of L and p

L and p are the two parameters that can be tuned in ExVQ. In this section, we evaluate how L and p affect the trade off between the classification accuracy and the VQ time. First, we evaluate the classification accuracy and VQ time when we change L with a fixed p. We change L from 5 to 15 (with step size 2) and test on one round of Caltech 101 with $p = 1/5, K = 256$. The classification accuracy and the VQ time are shown in Fig. 4a. When L is larger than 11, the VQ time is not reduced apparently. Besides, a larger L can induce a higher VQ error. However, all classification accuracies are similar with different L. When L is even larger, the VQ time cost will be bigger and the classification accuracy will drop.

Next, we evaluate ExVQ with different exclusion portion p with a fixed L. We change $1/p$ from 3 to 11 (with step size 2) and fix $L = 10$ on one round of Caltech 101 with $K = 256$. The classification accuracy and the VQ time are shown in



(a) (b)

Fig. 4: (a) The classification accuracy and the VQ time with L and fixed p; (b) The classification accuracy and the VQ time with $1/p$ and fixed L.

Table 4: Comparison with other related methods on Caltech 101, $K = 1024$.

ANN methods	VQ time (s)	Accuracy (%)
ExVQ	280	65.17
Spill tree	291	63.87
LSH	3247	63.97
LLC	4817	65.77

Fig. 4b. When p is smaller, the training of linear classifiers in the tree will be more accurate. So the VQ error will be smaller. Meanwhile, in the leaf node of the ExVQ tree, the active search set will be larger, which leads to longer VQ time. However, the classification accuracies keep almost the same for different p. When p is even smaller, the VQ time will cost more and the accuracy can be a little better (the linear projection is more accurate).

4.4 ExVQ and other related methods

Finally, we compare ExVQ with more related methods including spill tree [15], LSH [18] and LLC [4]. We evaluate the VQ time and the classification accuracy of these methods on Caltech 101 with $K = 1024$. We use the same tree structure of ExVQ for spill tree: the same L and p. So we can compare their classification accuracy under equivalent VQ time. We use the E^{2}LSH package[2] to implement LSH, which is to search all near neighbors within a radius R of the query point. In order to use it in the BOV model, we search exhaustively from the returned results of LSH to get the nearest neighbor. In E^{2}LSH, we use multiple radius for near neighbor search to save the VQ time. For LLC, we use the published Matlab code[3] for testing. The results are shown in Table 4.

ExVQ has the shortest VQ time. Spill tree's VQ time is close, but its classification accuracy drops significantly. Meanwhile, the VQ error rate of spill tree (41% $\gg$ 30%, although the error ranks are small) is much more than that of ExVQ, which influences the classification accuracy. ExVQ has similar accuracy

[2] `http://www.mit.edu/~andoni/LSH/`
[3] `http://www.ifp.illinois.edu/~jyang29/LLC.htm`

with LLC, but is about 20 times faster. Note that this comparison is qualitative rather than quantitative, because different programming languages are used. We also want to point out that ExVQ uses only a single core, while LLC utilizes multiple cores of a six-core CPU. Since LLC also needs to search kNN, ExVQ can be used to accelerate it. LSH is effective in large scale NN search problems. However, it is not suitable to get the nearest neighbor from the codebook with $K = 1024$, which costs more time even than the exhaustive NN search. The VQ error of LSH is not benign: its histogram of error ranks has a long tail, and the maximum error rank is 997. LSH has a lower classification accuracy than ExVQ, which coincides with our conjecture about how VQ errors impact the classification.

5 Conclusions

We presented ExVQ, an approximate nearest neighbor (ANN) search method for vector quantization (VQ) in BOV models. ExVQ achieves 10x speedup than exhaustive search and 2x speedup than state-of-the-art ANN methods. The key insight in ExVQ is an exclusive strategy, which can safely reduce the search range. The exclusion's quality is guaranteed by learning an accurate classifier. Thus, it is more efficient than the commonly used inclusive strategy in ANN search. We also systematically evaluated how VQ errors affect the accuracy of BOV-based visual classification, and conclude that some VQ methods have benign VQ errors, which does not reduce the classification accuracy even with 30% VQ errors. On four benchmark datasets, ExVQ leads to indistinguishable accuracy rates with the exhaustive search method and the FLANN method.

Future works include the following issues. First, effective exclusive NN search on large databases (e.g., $K = 100,000$) will be investigated. Second, integration of multiple trees in ExVQ to further improve the NN search efficiency. Third, ExVQ can be extended to search NN using arbitrary distance metrics, e.g., histogram intersection kernel [28]. Finally, we want to work out a theoretical framework for analyzing the effect of VQ errors to classification.

Acknowledgement. This work was supported in part by the following grants: National Science Foundation of China grant (61001146) and Shanghai Pujiang Program (12PJ1404300).

References

1. Sivic, J., Zisserman, A.: Video google: A text retrieval approach to object matching in videos. In: ICCV. (2003) 1470–1477
2. Fei-Fei, L., Perona, P.: A bayesian hierarchical model for learning natural scene categories. In: CVPR. Volume 2. (2005) 524–531
3. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: CVPR. (2006) 2169–2178
4. Wang, J., Yang, J., Yu, K., Huang, T., Gong, Y.: Locality-constrained linear coding for image classification. In: CVPR. (2010) 3360–3367

5. Yang, J., Yu, K., Gong, Y., Huang, T.: Linear spatial pyramid matching using sparse coding for image classification. In: CVPR. (2009) 1794–1801
6. Koen E. A. van de Sande, Gevers, T., Snoek, C.G.M.: Empowering visual categorization with the GPU. IEEE Transaction on Multimedia **13** (2011) 60–70
7. Fei-Fei, L., Fergus, R., Perona, P.: Learning generative visual models from few training example: an incremental bayesian approach tested on 101 object categories. In: CVPR. (2004)
8. Muja, M., Lowe, D.G.: Fast approximate nearest neighbors with automatic algorithm configuration. In: VISAPP. Volume 1. (2009) 331–340
9. Bhattacharya, S., Sukthankar, R., Jin, R., Shah, M.: A probabilistic representation for efficient large scale visual tasks. In: CVPR. Volume 2. (2011) 2593–2600
10. Gersho, A., Gray, R.M.: Vector quantization and signal compression. Springer (1991)
11. Nistér, D., Stewénius, H.: Scalable recognition with a vocabulary tree. In: CVPR. Volume 2. (2006) 2161–2168
12. Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.: Lost in quantization: Improving particular object retrieval in large scale image databases. In: CVPR. (2008)
13. Lepetit, V., Fua, P.: Keypoint recognition using randomized trees. IEEE TPAMI **28** (2006) 1465–1479
14. Moosmann, F., Nowak, E., Jurie, F.: Randomized clustering forests for image classification. IEEE TPAMI **30** (2008) 1632–1646
15. Liu, T., Moore, A.W., Gray, A., Yang, K.: An investigation of practical approximate nearest neighbor algorithms. In: NIPS. (2004)
16. Marszalek, M., Schmid, C.: Constructing category hierarchies for visual recognition. In: ECCV. (2008)
17. Gao, T., Koller, D.: Discriminative learning of relaxed hierarchy for large-scale visual recognition. In: ICCV. (2011)
18. Datar, M., Immorlica, N., Indyk, P., Mirrokni, V.S.: Locality-sensitive hashing scheme based on p-stable distributions. In: Symposium on Computational Geometry. (2004) 253–262
19. Jegou, H., Douze, M., Schmid, C.: Improving bag-of-features for large scale image search. IJCV **87** (2010) 316–336
20. Weiss, Y., Torralba, A., Fergus, R.: Spectral hashing. In: NIPS 21. (2009) 1753–1760
21. Jegou, H., Douze, M., Schmid, C.: Product quantization for nearest neighbor search. IEEE TPAMI **33** (2011) 117–128
22. Li, L.J., Fei-Fei, L.: What, where and who? Classifying events by scene and object recognition. In: ICCV. (2007) 261–268
23. Quattoni, A., Torralba, A.: Recognizing indoor scenes. In: CVPR. (2009) 413–420
24. Wu, J., Rehg, J.M.: CENTRIST: A visual descriptor for scene categorization. IEEE TPAMI **33** (2011) 1489–1501
25. Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J.: LIBLINEAR: A library for large linear classification. JMLR **9** (2008) 1871–1874
26. Vedaldi, A., Zisserman, A.: Efficient additive kernels via explicit feature maps. IEEE TPAMI **34** (2012) 480–492
27. Wu, J.: Power mean SVM for large scale visual classification. In: CVPR. (2012) 2344–2351
28. Wu, J., Rehg, J.M.: Beyond the Euclidean distance: Creating effective visual codebooks using the histogram intersection kernel. In: ICCV. (2009) 630–637