Large-scale R-CNN with Classifier Adaptive Quantization

Ryota Hinami and Shin'ichi Satoh

The University of Tokyo, National Institute of Informatics
{hinami,satoh}@nii.ac.jp

Abstract. This paper extends R-CNN, a state-of-the-art object detection method, to larger scales. To apply R-CNN to a large database storing thousands to millions of images, the SVM classification of millions to billions of DCNN features extracted from object proposals is indispensable, which imposes unrealistic computational and memory costs.
Our method dramatically narrows down the number of object proposals by using an inverted index and efficiently searches by using residual vector quantization (RVQ). Instead of k-means that has been used in inverted indices, we present a novel quantization method designed for linear classification wherein the quantization error is re-defined for linear classification. It approximates the error as the empirical error with pre-defined multiple exemplar classifiers and captures the variance and common attributes of object category classifiers effectively.
Experimental results show that our method achieves comparable performance to that of applying R-CNN to all images while achieving a 250 times speed-up and 180 times memory reduction. Moreover, our approach significantly outperforms the state-of-the-art large-scale category detection method, with about a 40∼58% increase in top-K precision. Scalability is also validated, and we demonstrate that our method can process 100K images in 0.13 s while retaining precision.

Keywords: image retrieval, object detection, image indexing

1 Introduction

With the explosive increase in multimedia data in recent years, there is a growing demand for information retrieval from large image/video databases. Large-scale object retrieval is a well-researched task that is used in many applications. Its purpose is to retrieve specific objects in a large image database as quickly as possible (i.e., immediately). Most of the existing object retrieval methods exploit the bag-of-visual-words (BoVW) model, where objects are retrieved using local descriptor matching. This approach is suited to specific objects (e.g., buildings, logos), but not to objects of generic *categories* (e.g., cat, bicycle), because local descriptor matching cannot capture the variation within a category well enough. A popular object categorization is based on the deformable part model (DPM) [1, 2], which uses HOG features instead of local descriptors to better describe object appearances in combination with SVM. R-CNN [3] extends the DPM by using convolutional neural network (CNN) features and achieves a significant improvement in accuracy. However, performing immediate object category detection on a large image database has not been tried until quite recently.

Recently, Aytar and Zisserman [4] attempted a large-scale object category detection. Unlike the traditional sliding window approach, their method achieved immediate detection throughout a large scale dataset. They developed a sparse representation of a HOG classifier and a new mid-level image representation using a vocabulary of *classifier patches* and performed fast retrieval using an inverted file index. Although this method achieved immediate retrievals on large-scale image datasets, its accuracy was significantly lower than that of the original sliding window-based approach. Moreover, because the performance of object category detection has significantly improved recently, this method still has a large performance gap compared with state-of-the-art object category detection. Most of the other work on accelerating object detection such as [5–9] assume that images are unknown (pre-processing of images is not permitted) and their runtime grows linearly with the number of images, namely the method that can process 150 images per second takes over 10 minutes to detect objects from 100K images, which is far from large-scale immediate object detection. Although other studies [10, 11] have focused on fast object retrieval and detection simultaneously, both methods are based on BoVW models designed for specific object retrieval and are thus not suitted to object category detection.

The performance of object category detection has dramatically increased in the past few years. Previously, the sliding window-based method with a HOG-based classifier such as DPM [1, 2] was widely used. Recently though, R-CNN [3] and its enhancements [12–14] have shown significant performance boosts over conventional methods. The key ideas of R-CNN are (1) limiting candidates by using object proposal instead of the exhaustive search with sliding window, and (2) exploiting features learned by a DCNN, which results in high classification accuracy even with linear SVM. Our objective is to apply R-CNN to all images in a large database immediately. However, this is a very difficult problem even if pre-processing using a database is permitted, because the number of object proposals grows to the millions or even billions if the database stores thousands to millions of images. That is, to apply R-CNN to a large database, we must classify millions to billions of high-dimensional vectors with linear SVM, which imposes high computational and memory costs.

Saloua et al. [15] accelerated the prediction phase of linear SVM by using locality sensitive hashing (LSH) [16]. LSH compresses the original high-dimensional vectors into small binary codes. A linear SVM classifier is then approximated by using the Hamming distance between the hashed data and the hashed hyperplane of the classifier, which can be computed extremely quickly. Although it achieves both high levels of compactness and computational efficiency, it assumes that all features x are L_2-normalized ($\|x\| = 1$), and thus, information on the original unnormalized feature is lost. Although many work have investigated the scalable SVM such as [17], most of them focus on the acceleration of the training phase and there is few work that focus on the acceleration of the prediction (testing) phase.

Recent progress in approximate nearest neighbor (ANN) searches has made it possible to handle billions of vectors efficiently. Several approaches have been studied for selecting nearest neighbor candidates from large amounts of vectors quickly, including tree-based indexing (e.g., k-d tree [18] and FLANN [19]), and hashing represented by LSH [16]. Product quantization (PQ) [20] is a widely used approach to compress vectors

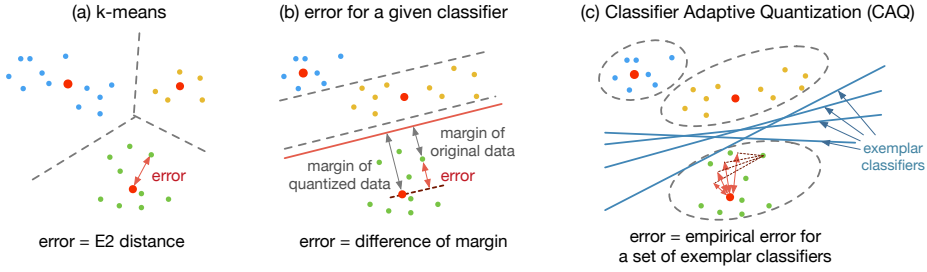

Fig. 1. An intuitive illustration of the difference in the quantization error between (a) the error defined by k-means and (b) the error in terms of a given classifier. (c) illustrates the error of CAQ, where red arrows indicate the error for each exemplar classifier (the error of CAQ is the sum of these).

into very short code so that all vectors can be stored in memory. PQ divides the feature space into a Cartesian product of subspaces that are quantized independently, which permits much finer quantization with an efficient learning procedure. PQ also permits efficient distance computations with a table lookup. Moreover, it has been shown to be more accurate than hashing-based methods such as spectral hashing [21] and hamming embedding [22].

An inverted index is another useful technique to avoid brute-force scans in large-scale searches. Inverted indices are built upon codewords learned from large datasets, and each codeword stores a list of data. This allows immediate access to a list of vectors close to any query vector and enables one to reduce the search time significantly over that of an exhaustive scan. The effectiveness of an inverted index for image retrieval was first demonstrated by Sivic and Zisserman in [23]. Jegou et al. in [20] also exploited an inverted index wherein the inverted list stores PQ-compressed data instead of the image IDs and performed well on billion-scale ANN searches. Most recent ANN search methods are based on this system, which is called IVFADC and combines an inverted index with coarse quantization and reranking based on compact code.

An inverted multi-index (IMI) [24, 25] is the current state-of-the-art indexing method of ANN searches; it is used instead of normal inverted indices. IMI achieves much finer subdivisions of the search space by constructing an inverted index using PQ. However, in our case, we find IMI does not improve the performance or even worse than IVFADC (k-means) in some cases (discussed in Sec. 4.4). Therefore, we use the standard inverted index and explore another novel quantizer suited to our task that improves performance without using IMI.

In this paper, we extend R-CNN to larger scales by incorporating state-of-the-art ANN search techniques (IVFADC). Our method narrows down a huge amount of feature vectors extracted from object proposals by using an inverted index and efficiently applies linear SVM by using fast distance computation with compressed code. We use residual vector quantization (RVQ) [26, 27] to compress data instead of PQ because

RVQ achieves significantly better performance than PQ does at a similar cost in our task.

In addition, we present a novel quantizer designed for linear classification to improve the performance of the inverted index. Most inverted indices methods utilize k-means, wherein the quantization error is defined as the Euclidean distance between a data point and its assigned codeword, and learns the codebook that minimizes the error. Although it is effective in a ANN search whose objective is to find the closest vector in Euclidean space, it is not suited to our task. We therefore re-define the quantization distortion that adapts a large-scale linear classification task. The basic idea is to define the quantization error as the difference in classification score before and after quantization as in Figure1 (b). However, this error depends on the classification boundary which, as indicated by the red line in Figure 1, is defined by each object category we want to detect, and is unknown in advance. We therefore prepare *exemplar classifiers*, a set of category classifiers trained beforehand and approximate the quantization distortion by their empirical error. This is equivalent to k-means with another metric that can capture the variation and essence of object classifiers, as shown in Figure 1 (c). We demonstrate that this quantization, called Classifier Adaptive Quantization (CAQ), is more effective than k-means in our task. It even outperforms IMI that is a state-of-the-art ANN search. We demonstrate our method—index inversion by CAQ and RVQ compression—can perform object detection immediately from a large database (takes 0.12 second from 100K images) while retaining the high accuracy of R-CNN, and thereby demonstrating that ANN search techniques are extremely effective at immediate large-scale object category detection.

Contributions. Our contributions are as follows: 1) a large-scale R-CNN architecture using the state-of-the-art techniques of an ANN search. It achieves excellent memory and computational efficiency while retaining the high level of accuracy of R-CNN, which itself outperforms a recent state-of-the-art large-scale object detection method by 40$\sim$58% in top-K precision; 2) classifier adaptive quantization, a novel quantization method for linear classification that performs better than traditional k-means in large-scale object detection tasks. To the best of our knowledge, no quantizations has been designed for linear classification.

2 Overview of Large-Scale R-CNN

In this section, we briefly overview our method. Our objective is to apply R-CNN [3] to all images in a large database immediately. R-CNN consists of the following steps: (i) detect object proposals by using a selective search; (ii) extract CNN features from the object proposals; (iii) apply SVM to the extracted features. In our setting, because the image database is given in advance, steps (i) and (ii) can be done offline. Therefore, our main focus is step (iii), i.e., applying SVM to a large amount of vectors efficiently.

The database consists of N images and around 2000 object proposals are detected in each image (n_i proposals from the ith image). $v_{i,j}$ ($i = 1, ..., N$, $j = 1, ..., n_i$) is a D-dimensional DCNN feature vector ($D = 4096$ in our case) extracted from each object proposal. SVM computes the classification score $S(v_{i,j}) = w \cdot v_{i,j} - b$, where w and b represent the hyperplane and bias of the SVM classifier. Since the total number of $v_{i,j}$s

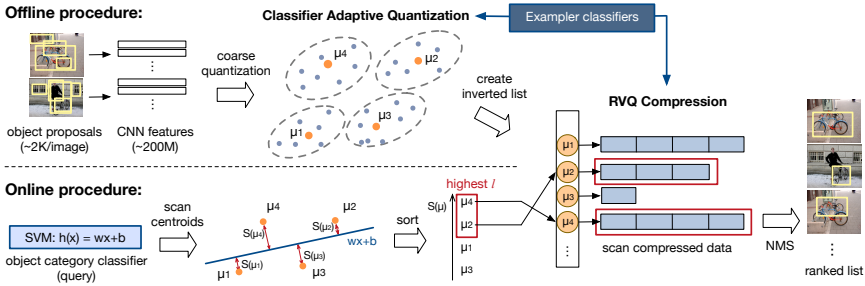

Fig. 2. Architecture of the proposed method.

is very large in a large database, computing the classification score for all vectors entails a huge number of computations and takes up a huge amount of memory. However, we can solve this problem by using an inverted index and reranking based on compact code.

2.1 Offline Procedure

Figure 2 is an overview of our method. Our approach can be divided into offline and online procedures. The offline procedure is performed as follows:

1. **Detect proposal objects and Extract features:** We follow a similar procedure to that of R-CNN. Object proposals are first detected in a selective search [28], entailing around 2000 proposals per image. Around $2000 \times N$ proposals are thus obtained from the image datasets. DCNN features are then extracted from each proposal.
2. **Construct an inverted index:** To avoid an exhaustive search, we construct an inverted index. We use a structure similar to the one proposed in [20]. Feature vectors $v_{i,j}$ are quantized coarsely into k clusters represented by k codewords $c(1), ..., c(k)$ (k-means is used in [20]). An inverted index is then constructed with k lists $L_1, ..., L_k$, where each list L_i stores data that belongs to the corresponding cluster with a codeword $c(i)$. Instead of k-means, we use a novel quantization designed for linear classification (see Sec. 3). We call this Classifier Adaptive Quantization (CAQ).
3. **Compress data with residual vector quantization:** To reduce memory and computational costs, feature vectors are stored in an inverted file structure after compressing them into small codes. We take a product quantization (PQ)-based approach [20]. Although it is mainly used for Euclidean distance approximations, it can be easily extended to approximations of inner products. We tested various compression method and found that residual vector quantization (RVQ) [26, 27] has significantly better performance than PQ or optimized product quantization (OPQ) that is used in IVFADC and IMI. Figure 3 shows a comparison of object detection performance with compressed features. It also indicates the number of sub-codebooks M should be large enough (M=32 to 128) to achieve sufficient object detection performance with compressed codes.

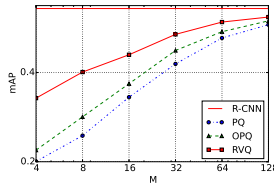

Fig. 3. Comparison of data compression methods on the PASCAL VOC 2007 object detection task. R-CNN calculates detection scores for all object proposals by applying SVM to the original DCNN features. The detection procedures of PQ, OPQ, and RVQ are also based on R-CNN, but approximate the SVM score by using compressed codes.

RVQ learns multiple sub-codebooks one by one by minimizing the error greedily. The first sub-codebook is learned by performing k-means on original vectors. The second sub-codebook is learned by performing k-means on residual vectors with respect to the assigned codewords of the first sub-codebook, which is a similar process to the residual encoding used in IVFADC. This process is repeated until all sub-codebooks are learned. RVQ is also related to additive quantization (AQ), where the vector is approximated by the sum of codewords from different codebooks. However, encoding of AQ using such and such a method becomes too slow as the number of codebooks grows, and this makes it impossible to compress a large number of high-dimensional CNN features. Therefore, we decided to use RVQ with M=64.

We do not encode the residual vector with respect to the coarse quantizer as in IVFADC[20] because we found that residual encoding does not improve performance. This suggested that the quantization of CNN features has to be extremely fine in order to achieve reasonable performance (M=64), and therefore, the effect of coarse quantization is more limited than in the original IVFADC (M=$\sim$16 is used).

2.2 Online Procedure

Next, we describe the online procedure of the proposed architecture. We assume that an SVM classifier $h(x) = w_q \cdot x + b_q$ is given as query. We scan an inverted file structure by using the following procedure:

1. **Scan the codewords of the coarse quantizer:** Query SVM is applied to codewords of an inverted index; i.e., a classification score $S(c(i)) = w_q \cdot c(i) + b_q$ is computed for each codeword $c(i)(i = 1, ..., k)$.
2. **Scan the data in list with the highest score:** The codewords with l highest score $c(i_1), c(i_2), ..., c(i_l)$ are selected, and their corresponding inverted lists $L_{i_1}, L_{i_2}, ..., L_{i_l}$ are scanned. The inner product $w_q \cdot v$ is approximated by the inner product between the query and RVQ-compressed data, i.e., $w_q \cdot v \sim w_q \cdot \sum_{m=1}^{M} c_m(i_m) \sim \sum_{m=1}^{M} w_q \cdot c_m(i_m)$, where $c_m(i_m)$ is the i_mth codeword of the mth sub-codebook

assigned to v. This can be computed very quickly using a pre-computed distance table. The table is created in a manner similar to the asymmetric distance computation (ADC) in [20]

3. **Perform non-maximum suppression and output results:** The scanned candidates are sorted in the order of the computed score, and a ranked list is output after non-maximum suppression (NMS). NMS rejects regions that overlap with a higher scoring region and have an intersection-over-union (IoU) value of > 0.3. Note that if only the top-K scoring objects have to be detected, NMS is applied to only the top-ranked regions.

3 Classifier Adaptive Quantization

3.1 Quantization Distortion for Linear Classifier

In this section, we describe a quantization method suited to linear classification. K-means is a popular vector quantization method that is optimal for some tasks including a nearest neighbor search by minimizing the quantization distortion. It defines the quantization distortion as:

$$E = \frac{1}{n} \sum_{x} \|x - c(i_x)\|^2, \tag{1}$$

where $\|\cdot\|$ denotes the l_2-norm, n is the total number of data samples, i_x is a codeword ID assigned to data x, and the i_xth codeword is denoted as $c(i_x)$. K-means learns the codebook minimizing this distortion.

Although k-means is suitable for nearest neighbor searches wherein the error is defined by the distance between a query and a data point in a Euclidean space, we assert that it is not optimal for linear classification. In our case, through the quantization process, the classifier score of the original data is approximated by the score of the quantized data. Therefore, it is natural that the quantization error is defined by the difference in the classifier score before and after quantization, as illustrated in Figure 1. Here, we denote the hypothesis of a certain linear SVM classifier as $h(x) = w \cdot x + b$. We can define the quantization distortion for this classifier as follows:

$$E = \frac{1}{n} \sum_{x} \|w \cdot x - w \cdot c(i_x)\|^2, \tag{2}$$

where the bias b is canceled by taking the difference and does not affect the quantization distortion at all. Although this distortion can be defined if the classifier is known in advance, the classifier is often given as a query in a retrieval task, and hence would be unknown before it is given. We thus modify this quantization distortion to be able to deal with unknown classifiers.

We prepare a set of SVM classifiers trained on a variety of categories ($180 \sim 2000$ classifiers in our case). We call them *exemplar classifiers*. We approximate the distortion for any query classifier by the empirical error of these exemplar classifiers. We assume they capture common attributes in any category of classifiers, as well as their variance.

The training of exemplar classifiers is detailed in Sec. 4.1. Let us denote the hyperplanes of exemplar classifiers as $\{w^{(1)}, w^{(2)}, ..., w^{(n_w)}\}, w^{(i)} \in \mathbb{R}^D$, where n_w is the number of exemplar classifiers, and these form the rows of a matrix $W \in \mathbb{R}^{n_w \times D}$. We define the empirical distortion of exemplar classifiers as follows:

$$E = \frac{1}{n} \sum_x \|Wx - Wc(i_x)\|^2, \tag{3}$$

which is the distortion of the *classifier adaptive quantization* (CAQ) we present. By minimizing E, we can expect reasonable performance for all possible object classifiers. Furthermore, we use it below to formulate the encoding and codebook learning.

3.2 Encoding and Codebook Learning

In this section, we formulate the encoding and codebook learning of CAQ and show that they can be performed by revising the k-means algorithm with little additional cost. First, let us consider the task of encoding, i.e., finding the codeword assigned to a data sample x. An assignment i_x that minimizes the coding error is obtained by solving the following equation:

$$i_x = \arg\min_i \|Wx - Wc(i)\|^2, \tag{4}$$

which is equivalent to doing a nearest neighbor search to data projected by W.

Next, let us show how to learn a codebook that minimizes the distortion in Eq.(3). Similarly to k-means, we take an alternating optimization strategy wherein the codebook and assignment are optimized alternately. The minimization over the assignment i with a fixed codebook is performed by solving Eq.(4). The following equation is used to update a codebook given an assignment:

$$c(i) = \arg\min_{c(i)} \sum_{x \in N_i} \|Wx - Wc(i)\|^2, \tag{5}$$

where N_i is the set of data samples assigned to codeword $c(i)$. If $c(i)$ is the centroid of the cluster ($c(i) = \frac{1}{n_i} \sum_{x \in N_i} x$, where n_i is the size of N_i), Eq.(5) is satisfied. From Eq.(4) and Eq.(5), we can see that this problem is equivalent to the k-means if data samples are Wx and codewords are $Wc(i)$. Therefore, assignments can be obtained simply by applying k-means in the subspace into which the data samples are projected by W, which is equivalent to the k-means with Mahalanobis distance. Since the codewords are defined in a data space, they can be obtained as the centroids of the data assigned to each cluster.

CAQ is based on the simple k-means algorithm, which has two merits. Firstly, we can make use of the rich literature on k-means and distributed implementations, e.g., clustering with large datasets and vocabularies can be efficiently performed using fast k-means such as [29]. Secondly, CAQ is easily incorporated into other systems based on k-means, such as joint inverted indexing [30]. Note that although W can be generated with other metric learning approaches, we at least confirmed that CAQ outperformed the Mahalanobis metrics. This is true because the CAQ metric considers the bias of the object category classifier while the Mahalanobis metric considers only the distribution of the data.

4 Experiments

4.1 Datasets and Implementation Details

We used the popular PASCAL VOC 2007 detection dataset to evaluate our method. The test set consisted of 4952 images from 20 different categories. In addition, the validation set of ILSVRC 2011 and 2012 [31] (100K images in total) were used as distractors in the large-scale experiments. A previous study [4] also used these distractors to evaluate large-scale category detection.

Our implementation was based on R-CNN. We used the same network as R-CNN, which was pre-trained for ImageNet classification and fine-tuned for PASCAL VOC object detection. We used 4096-dimensional fc7 feature vectors. Different from R-CNN, our method is designed for large-scale data, as explained in Sec. 2; data is accessed through an inverted index and each inverted list stores data compressed by RVQ. To learn the codebook, we used randomly selected regions detected with a selective search. We used 500000 samples to learn a codebook for coarse quantization (an inverted index) and 100000 samples for the quantization of the data compressions (PQ, OPQ and RVQ). PQ, OPQ, and RVQ set K=256 as the size of the sub-codebook (8 bits assigned per sub-codebook) in all experiments, while k=4096 and 16384 are used for coarse quantization. We used hard negative mining to train the SVMs used in the test phase (the same training algorithm and hyper-parameters as in the original R-CNN). The training and validation parts of the PASCAL VOC 2007 (5011 images in total) were used to train the SVMs and learn the codebooks.

In addition, we prepared exemplar classifiers for CAQ. Here, we describe how we constructed these exemplar classifiers W. The training set of the ILSVRC 2014 detection datasets, which has 200 categories, was used to train detectors. This set is completely separated from the test datasets. We constructed four sets of exemplar classifiers, W_{180}, W_{200}, W_{1800}, and W_{2000}. W_{200} was formed from 200 classifiers trained on 200 categories of the ILSVRC training set. W_{180} excluded the 20 categories corresponding to PASCAL VOC from these 200 categories. W_{2000} and W_{1800} were constructed from 200 and 180 categories of ILSVRC; ten detectors were constructed from each category. For each category, 10 sets of 500 positive samples were randomly selected from the training set of ILSVRC and they formed 10 detectors (one detector per set). W_{200} and W_{2000} were not used in most of our experiments in order to strictly reproduce the situation that the query is completely unknown during training.

In addition to these four sets, we used *eigen queries* as exemplar classifiers (introduced in [32]). We generated eigen queries from W_{1800}. Here, we performed eigenvalue decomposition on the covariance matrix of W_{1800} and selected eigenvectors corresponding to the largest d eigen values to be eigen queries that were the principal components of W_{1800}.

4.2 Comparison with R-CNN

We first demonstrate that our method compares well with R-CNN on the PASCAL VOC 2007 dataset. We compared R-CNN with three versions of our method: (i) an exhaustive search where all feature vectors compressed by RVQ are scanned (referred

to as RVQ), (ii) a non-exhaustive search where an inverted index is constructed by CAQ and inverted lists store the original 4096-dimensional vectors (referred to as CAQIVF + original), and (iii) a non-exhaustive search where inverted lists store RVQ-compressed vectors (referred to as CAQIVF + RVQ). M=64 (64 bytes code) was used in RVQ, and k=16384 and l=64 were used in the inverted file retrieval. We used W_{1800} as the exemplar classifiers for CAQ. Note that *R-CNN* which is used as baseline in this paper corresponds to an exhaustive search to the original 4096-dimensional vectors, and other settings (e.g., feature extraction are done offline) are the same as our method.

Table 1 shows the average precision on PASCAL VOC 2007 for our three versions and R-CNN as a baseline. Our methods compare well with R-CNN; even the non-exhaustive retrieval RVQ compression (CAQIVF + RVQ) achieved a mAP of 50.1%, while R-CNN obtained a mAP of 54.2%. Next, we evaluated the efficiency. Table 2 lists the number of code comparisons, timings, and amount of memory used. The number of code comparisons and the timings are the means over PASCAL's 20 categories. The results for CAQIVF + RVQ show the first scan for codewords of the inverted index and the second scan for compressed vectors separately. RVQ achieved a 90× speed-up and 256× memory reduction compared with R-CNN, which demonstrates the efficiency of RVQ compression. CAQIVF + RVQ reduced the number of comparisons from about 10M to 42K, including both codewords and compressed data, which led to a 250× faster (and 180× memory efficient) search compared with R-CNN. An inverted index is more effective on a larger database, as we show later in Sec. 4.6. Note that the timings in Table 2 exclude the NMS processing, because it depends on the way of evaluation. We found that NMS takes 0.1 s to detect all objects on PASCAL VOC 2007 and takes ∼10 ms to detect the top-100 scored objects. It should be noted that our method is compatible with Fast and Faster R-CNN and offline processing time for feature extraction can be reduced by using these methods.

Table 1. Comparison of our method with R-CNN on PASCAL VOC 2007 (%). M=64, k=16384, and l=64 were used for our methods.

VOC 2007	aero	bike	bird	boat	bott	bus	car	cat	chair	cow	table	dog	horse	mbik	pers	plant	sheep	sofa	train	tv	mAP
R-CNN	64.3	69.6	50.1	41.9	32.1	62.6	70.9	60.9	32.7	58.5	46.2	56.1	60.4	67.2	54.1	31.5	52.8	48.9	57.8	64.8	54.2
RVQ(Exhaustive)	62.8	66.6	46.5	38.0	29.0	62.2	68.5	59.5	26.1	54.8	45.3	49.9	56.3	66.0	53.1	28.5	52.9	45.5	52.2	62.3	51.3
CAQIVF+original	61.1	67.8	48.3	40.0	32.5	62.4	67.9	57.9	28.6	57.1	44.9	55.3	58.4	66.4	47.9	30.8	53.4	47.5	56.1	64.4	52.4
CAQIVF+RVQ	60.3	65.6	44.9	36.7	28.9	60.7	66.9	56.8	25.0	53.0	45.7	49.1	55.5	65.6	47.0	27.2	52.4	44.9	50.9	61.8	50.0

4.3 Evaluation of CAQ

In this section, we make a detailed evaluation of CAQ. We constructed an inverted file structure by using coarse quantization with CAQ. For pure evaluations of the performance of CAQ as a quantizer of an inverted index, we measured the recall without RVQ re-ranking. Recall is defined as the number of relevant objects in a list returned by the inverted index divided by the total number of ground truth objects. The performance

Table 2. Number of code comparisons, search time, and memory consumption in PASCAL VOC 2007 test (4952 images). The accuracy of the corresponding methods are shown in Table 1. The times were measured on a single core.

method	comparisons	search time	memory
R-CNN	9927228	6258.5 ms	163 GB
RVQ	9927228	69.5 ms	0.64 GB
CAQIVF + original	41892	518.0 ms	163 GB
CAQIVF + RVQ			
inverted index	16384	15.0 ms	0.27 GB
compressed	25508	6.5 ms	0.64 GB
total	41892	24.5 ms	0.91 GB

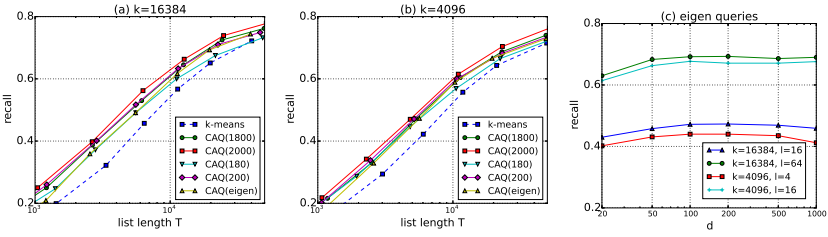

Fig. 4. Evaluation of CAQ on PASCAL VOC 2007. (a), (b): Recall as a function of the candidate list length. CAQ (eigen) used eigen queries with d=180. The codebook size is k=16384 in (a) and k=4096 in (b). (c): using eigen queries as exemplar classifiers while varying their number d from 20 to 1000.

of k-means was also measured as a baseline. To investigate the effect of exemplar classifiers, we tested four different cases: W_{180}, W_{200}, W_{1800}, and W_{2000}. We also used eigen queries as exemplar classifiers while varying the number of eigenvectors d.

Figure 4(a), (b) show the recall as a function of the list length, i.e., the number of codes returned by an inverted index. We used codebook sizes of k=16384 in (a) and k=4096 in (b) and varied l, the number of lists to be scanned, from 1 to 128. All our methods using CAQ outperformed k-means, which demonstrates the superiority of CAQ over k-means as an inverted index of this task. The performance on W_{200} (W_{2000}) was higher than on W_{180} (W_{1800}). This indicates that the accuracy depends on whether exemplar classifiers include the same category as the query. In addition, W_{1800} (W_{2000}) was higher than W_{180} (W_{1800}). It appears that exemplar classifiers with a larger number of classes can better capture the essence of the category classifier. The eigen queries method with $d = 180$ was inferior to W_{1800}, but superior to W_{180} in most case, especially with a larger list length. The results of eigen queries for various d are shown in Figure 4(c). The highest recall was on $d = 100, 200$, although the value is not so sensitive to changing d. Although accuracy of the eigen queries was not so good, it can reduce the computational cost of encoding and learning the codebook, an effect that stands out when the number of exemplar classifiers is large.

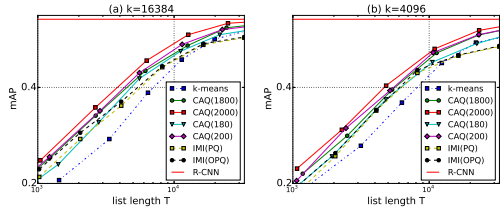

Fig. 5. Object detection performance of CAQ, k-means, and IMI with PQ and OPQ on PASCAL VOC 2007. The figure plots mAP as a function of the candidate list length. The codebook size is k=16384 in (a) and k=4096 in (b).

4.4 Comparison of Object Detection Performance

In this section, we compare the object detection performances of various indexing methods. We measured mAP and changed the inverted index as follows: CAQ with various exemplar classifiers, k-means, and IMI [24, 25] using PQ and OPQ. IMI was adapted for our task by changing the order of the priority queue to the score of the linear classifier. IMI was constructed using PQ and OPQ with two sub-codebooks. We performed RVQ-based re-ranking and NMS on a list of candidates returned by the inverted index to evaluate the performance of the overall object detection pipeline.

Figure 5 shows the performance comparisons for codebook sizes (sub-codebook sizes in IMI) of k=16384 in (a) and k=4096 in (b). CAQ outperformed k-means, IMI (PQ), and IMI (OPQ) in most cases. Note that IMI was inferior to even the k-means-based inverted index with larger list length despite that it has been shown to outperform IVFADC (k-means) in ANN searches. This implies that a finer partition is not always a good thing, which has a different nature from that of a nearest neighbor search. In our task, relevant regions sometimes have lower scores than irrelevant regions do because the classifier (SVM + CNN feature) is not perfect, due to the so-called semantic gap [33]. By coarse quantization, in our case, regions with similar appearances tend to be clustered together and relevant regions with lower scores can be detected together along with other many relevant regions with higher scores. Conversely, isolated regions (not similar to relevant objects) that have high scores but are irrelevant can be excluded by the inverted index. We consider these are the reasons that a coarser inverted index sometimes outperformed a finer one. Thus, we consider that a moderate-grained quantizer in which objects of similar categories belong to the same cluster is the best as an inverted index used in this task. From this perspective, CAQ is certainly reasonable because it exploits the response to the various object classifiers, and thus, objects that are semantically close in terms of object category tend to belong to the same cluster. This seems to be the reason that the CAQ-based inverted index outperformed the k-means one and IMI.

4.5 Comparison with State-of-the-art Methods

To show the superiority of our method over the previous ones, we compared it with [4], a recently proposed method of large-scale category detection. We also compared it

with Video Google [23], which is based on the bag-of-visual-words (BoVW) method. Following [4], we used the top-k precision (k=10, 50, and 100) to evaluate them. We evaluated RVQ, CAQIVF + original, and CAQIVF + RVQ using the same settings as in Sec. 4.2. NMS was applied to the top-scored regions until the number of candidates reached 100.

Table 3 shows the precisions of the top 10, 50, and 100 ranked detections on PASCAL 2007. Included are the results from [4] (FS + HOG-SC and FS + RR + HOG-SC) and for Video Google (also reported in [4]). FS + HOG-SC uses an inverted index and re-ranking that re-evaluates candidates using the original HOG classifier template. FS + RR + HOG-SC adds a second re-ranking stage similar to what is done in NMS. Video Google was tested with two different vocabulary sizes, 10K and 200K. The results show that our method significantly outperforms [4] and Video Google. Compared with FS + RR + HOG-SC, CAQIVF + RVQ increases by 57.5%, 53.0%, and 40.0% on PR@100, PR@50 and PR@10 in mean precision of 20 categories. There are mainly two reasons for this significance performance boost: 1) R-CNN was more accurate than DPM, 2) the proposed method was comparable in accuracy to the R-CNN due to its used of state-of-the-art ANN search techniques and CAQ, while [4] performed far worse than the original DPM. The search time was also shorter. CAQIVF + RVQ can processed the PASCAL VOC 2007 testset in 38.2 ms including the processing time of NMS, while [4] reported that FS + RR + HOG-SC takes 1.3 s. Note that although these timings were not measured in completely the same experimental environment, both methods were evaluated using a single core. Moreover, we can see, by comparing the R-CNN with CAQIVF + original or by comparing RVQ with CAQIVF + RVQ, that a non-exhaustive search does not decrease performance at all. Rather, CAQIVF + RVQ outperformed RVQ. This indicates an inverted index is especially effective in retrieval tasks that only need to detect top-ranked objects.

Table 3. Comparison with previous methods on PASCAL VOC 2007(%). Mean values of top 10, 50, and 100 precisions over 20 categories are shown. The search times of our methods include the time taken by the whole procedure including the NMS re-ranking. Our versions used M=64, k=16384, and l=64. Although times of FS + HOG-SC and FS + RR + HOG-SC reported in [4] are also shown in brackets for reference, they are not directly comparable because the experimental environment was different.

method	PR@10	PR@50	PR@100	search time
R-CNN	96.5	90.8	87.0	6.2 s
RVQ	91.0	89.8	82.9	77.4 ms
CAQIVF + original	97.0	90.9	85.2	529.9 ms
CAQIVF + RVQ	93.5	89.8	85.0	29.1 ms
FS + HOG-SC [4]	45.9	28.9	22.0	(1.2 s)
FS + RR + HOG-SC [4]	53.5	36.8	27.5	(1.3 s)
Video Google 10K [23]	25.8	14.8	10.2	-
Video Google 200K [23]	32.9	17.4	11.1	-

4.6 Large-scale Experiments

To investigate the scalability of our method, we added 100K distractors from the ILSVRC 2011/2012 validation sets to the PASCAL VOC 2007 dataset (105K images in total). Since we do not have reliable ground truth for the ILSVRC validation set, we performed a manual evaluation similar to what is done in [4]. In these experiments, we extracted features using the Fast R-CNN framework [13], an accelerated version of R-CNN, to speed-up feature extraction. We used the VGG-16 network [34] distributed by one of the authors of [13]. It was trained for detection on PASCAL VOC 2007 and used 4096-dimensional fc6 features. SVM was also trained on these features. We used the CAQIVF + RVQ with the parameters of M=64, k=16384, and l=16 in this experiment.

Table 4 shows the results for PASCAL VOC 2007 alone and the combined set. We can see that the performance is not affected by the distractors. Moreover, the search time was only 130 ms while the exhaustive search of RVQ took 2.7 s.

Table 4. Large-scale experiments with 100K added distractors. k=16384, l=16, and M=64 were used.

Datasets	PR@10	PR@50	PR@100	search time	memory
PASCAL (5K)	94.0	87.6	81.1	29.1 ms	0.91 GB
PASCAL+ILSVRC (105K)	91.0	88.0	81.2	129.6 ms	13.61 GB

5 Conclusion

This paper introduced large-scale R-CNN. The proposed method achieved immediate and accurate object category detection from a large image database by combining the state-of-the-art object detection method and state-of-the-art nearest neighbor search. Our experiments demonstrated that this method performed comparably to the original R-CNN on PASCAL VOC 2007 in terms of accuracy, but with a 250 times speed-up and 180 times memory reduction. Moreover, it significantly outperformed the state-of-the-art large-scale category detection method, and its accuracy and search speed were not affected by the addition of 100K distractors.

We also presented classifier adaptive quantization (CAQ), whose quantization distortion is defined on the basis of the linear classification score to further improve the performance of our large-scale R-CNN. We confirmed that performance significantly increased as a result of using CAQ as an inverted index instead of k-means and that it also outperformed an inverted multi-index.

References

1. Felzenszwalb, P., Mcallester, D., Ramanan, D., Irvine, U.C.: A Discriminatively Trained , Multiscale , Deformable Part Model. In: CVPR. (2008)

2. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part-based models. PAMI **32**(9) (sep 2010) 1627–45
3. Girshick, R., Donahue, J., Darrell, T., Berkeley, U.C., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR. (2014)
4. Aytar, Y., Zisserman, A.: Immediate , scalable object category detection. In: CVPR. (2014)
5. Dean, T., Ruzon, M.a., Segal, M., Shlens, J., Vijayanarasimhan, S., Yagnik, J.: Fast, accurate detection of 100,000 object classes on a single machine. In: CVPR. (2013)
6. Sadeghi, M., Forsyth, D.: 30Hz Object Detection With Dpm V5. ECCV (2014) 65–79
7. Girshick, R.: Fast R-CNN. (2015)
8. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. Nips (2015) 1–10
9. Redmon, J., Girshick, R., Farhadi, A.: You Only Look Once: Unified, Real-Time Object Detection. CVPR (2015)
10. Lampert, C.H.: Detecting objects in large image collections and videos by Efficient Subimage Retrieval. In: ICCV. (2009)
11. Shen, X., Lin, Z., Brandt, J., Avidan, S., Wu, Y.: Object retrieval and localization with spatially-constrained similarity measure and k -NN re-ranking. In: CVPR. (2012)
12. He, K., Zhang, X., Ren, S., Sun, J.: Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. In: ECCV. (2014)
13. Girshick, R.: Fast r-cnn. In: ICCV. (2015)
14. Ouyang, W., Wang, X., Zeng, X., Qiu, S., Luo, P., Tian, Y., Li, H., Yang, S., Wang, Z., Loy, C.c., Tang, X.: DeepID-Net : Deformable Deep Convolutional Neural Networks for Object Detection. In: CVPR. (2015)
15. Litayem, S., Joly, A., Boujemaa, N.: Hash-Based Support Vector Machines Approximation for Large Scale Prediction. In: BMVC. (2012)
16. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In: FOCS. Volume 51. (2006) 459–468
17. Akata, Z., Perronnin, F., Harchaoui, Z., Schmid, C.: Good practice in large-scale learning for image classification. IEEE Transactions on Pattern Analysis and Machine Intelligence **36**(3) (2014) 507–520
18. Silpa-anan, C., Hartley, R.: Optimised KD -trees for fast image descriptor matching. In: CVPR. (2008)
19. Muja, M., Lowe, D.G.: Fast Approximate Nearest Neighbors with Automatic Algorithmic Configuration. In: VISAPP. (2009)
20. Jégou, H., Douze, M., Schmid, C.: Product Quantization for Nearest Neighbor Search. PAMI **33**(1) (2011) 117–128
21. Weiss, Y., Torralba, A., Fergus, R.: Spectral Hashing. In: NIPS. (2009)
22. Jegou, H., Douze, M., Schmid, C.: Hamming embedding and weak geometric consistency for large scale image search. In: ECCV. (2008)
23. Sivic, J., Zisserman, A.: Video Google: a text retrieval approach to object matching in videos. In: ICCV. (2003)
24. Babenko, A., Lempitsky, V.: The Inverted Multi-Index. In: CVPR. (2012)
25. Babenko, A., Lempitsky, V.: The Inverted Multi-Index. PAMI **37**(6) (2015) 1247–1260
26. Chen, Y., Guan, T., Wang, C.: Approximate nearest neighbor search by residual vector quantization. Sensors **10**(12) (2010) 11259–11273
27. Juang, B.H., Gray Jr, A.H.: Multiple stage vector quantization for speech coding. In: Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'82. Volume 7., IEEE (1982) 597–600
28. Uijlings, Jasper RR and van de Sande, Koen EA and Gevers, Theo and Smeulders, A.W.: Selective Search for Object Recognition. IJCV **104**(2) (2013) 154–171

29. Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.: Object retrieval with large vocabularies and fast spatial matching. In: CVPR. (2007)
30. Xia, Y., He, K., Wen, F., Sun, J.: Joint inverted indexing. In: ICCV. (2013)
31. Deng, J., Berg, A., Satheesh, S., Su, H., Khosla, A., Fei-Fei, L.: Imagenet large scale visual recognition competition 2012 (ilsvrc2012) (2012)
32. Raval, N., Tonge, R.V., Jawahar, C.V.: Image retrieval using eigen queries. In: ACCV. (2013)
33. Smeulders, A., Worring, M.: Content-based image retrieval at the end of the early years. PAMI **22**(12) (2000) 1–32
34. Simonyan, K., Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition. In: ICLR. (2015)