Classemes: A Compact Image Descriptor for Efficient Novel-Class Recognition and Search

Lorenzo Torresani, Martin Szummer, and Andrew Fitzgibbon

Abstract. In this chapter we review the problem of object class recognition in large image collections. We focus specifically on scenarios where the classes to be recognized are not known in advance. The motivating application is "object-class search by example" where a user provides at query time a small set of training images defining an arbitrary novel category and the system must retrieve images belonging to this class from a large database. This setting poses challenging requirements on the system design: the object classifier must be learned efficiently at query time from few examples; recognition must have low computational cost with respect to the database size; finally, compact image descriptors must be used to allow storage of large collections in memory. We review a method that addresses these requirements by learning a compact image descriptor – *classemes* – yielding good categorization accuracy even with efficient linear classifiers. We also study how data structures and methods from text-retrieval can be adapted to enable efficient search of an object-class in collections of several million images.

1 Introduction

The accuracy of object category recognition is improving rapidly, particularly if the goal is to retrieve or label images where the category of interest is the primary subject of the image. However, existing techniques do not scale well to searching in large image collections. This chapter identifies three requirements for such scaling, and describes representations and retrieval methods that satisfy them.

Lorenzo Torresani
Dartmouth College, 6211 Sudikoff Lab, Hanover, NH 03755, U.S.A.
e-mail: lorenzo@cs.dartmouth.edu

Martin Szummer · Andrew Fitzgibbon
Microsoft Research, 7 JJ Thomson Avenue, Cambridge, CB3 0FB, U.K.
e-mail: {szummer,awf}@microsoft.com

R. Cipolla et al. (eds.), *Registration and Recognition in Images and Videos*,
Studies in Computational Intelligence 532,
DOI: 10.1007/978-3-642-44907-9_5, © Springer-Verlag Berlin Heidelberg 2014

1. Interesting large-scale applications must support recognition of **novel categories**. This means that a new category can be presented as a set of training images, and a classifier learned from these new images can be run efficiently against the large database. Depending on the application, the user may define the query category either by supplying a set of image examples of the desired class, by performing relevance feedback on images retrieved for predefined tags, or perhaps by bootstrapping the recognition via text-to-image search [12]. In all these cases, the classifiers cannot be precomputed during an offline stage and thus both training and testing must occur efficiently at query-time in order to be able to provide results in reasonable time to the user.
2. Large-scale recognition benefits from a **compact descriptor** for each image, for example allowing databases to be stored in memory rather than on disk.
3. The ideal descriptor also provides good results with **linear classifiers**, such as linear SVMs, or tf-idf rankers [24], as these can be evaluated efficiently even on large databases.

Although a number of systems satisfy these desiderata for recognition of specific object-instances [27, 17], places [6] and whole scenes [43], we argue that these requirements cannot be addressed by traditional systems in the context of object-category recognition. This is due to the large computational and storage complexities of modern object-classifiers, which rely on high-dimensional image descriptors and expensive non-linear decision functions. For example, the current state-of-the-art in categorization is represented by multiple kernel combiners, such as the LP-β classifier [13], which compute non-linear (kernel-based) functions of multiple low-level features. These nonlinearities are critically necessary to achieve good classification accuracy: for example, compare in figure 2 the difference in accuracy between LP-beta13 and Xsvm, which represent, respectively, a multiple kernel combiner and a linear SVM trained on the same combination of low-level features. However, kernel-based classifiers cannot be used in our search setting, since the classes to recognize are not known at the time of the creation of the database and thus the kernel-distances cannot be precomputed: novel-class recognition with non-linear models would require evaluating the kernel-distance between each database image and (a subset of) the training images provided at query-time, which clearly cannot be accomplished in the real-time demanded by a search application. Furthermore, the multiple, high-dimensional image descriptors needed by LP-β would pose challenging storage requirements for large databases.

In this chapter we describe a system that addresses these requirements by using multiple-kernel combiners as an image representation instead of as a classification model: the idea is to use an image descriptor containing as entries the outputs of a set of *predefined* category-specific classifiers applied to the image. Because these basis-classifiers provide a rich coding of the image, simple *linear* models (e.g., linear SVMs) trained on this representation can approach state-of-the art accuracy, satisfying the requirements listed above. The obvious (but only partially correct) intuition is that a novel category, say `duck`, can be effectively expressed in terms of the outputs of the basis-classifiers (which we refer to as "classemes"), describing either objects similar to ducks, or objects seen in conjunction with ducks.

Table 1 Highly weighted classemes. Five classemes with the highest LP-β weights for the retrieval experiment, for a selection of Caltech256 categories. Some may appear to make semantic sense, but it should be emphasized that our goal is simply to create a useful feature vector, not to assign semantic labels. The somewhat peculiar classeme labels reflect the ontology used as a source of base categories.

New category	Highly weighted classemes				
cowboy-hat	helmet	sports_ track	cake_ pan	collectible	muffin_ pan
duck	bomber_ plane	body_ of_ water	swimmer	walking	straight
elk	figure_ skater	bull_ male_ herd_ animal	cattle	gravesite	dead_ body
frisbee	watercraft_ surface	scsi_ cable	alarm_ clock	hindu	serving_ tray
trilobite-101	convex_ thing	mined_ area	cdplayer	roasting_ pan	western_ hemisphere_ person
wheelbarrow	taking_ care_ of_ something	baggage_ porter	canopy_ closure_ open	rowing_ shell	container_ pressure_ barrier

In practice, the reason this descriptor will work is slightly more subtle. It is not required or expected that these base categories will provide useful semantic labels, of the form `water`, `sky`, `grass`, `beak`. On the contrary, the assumption is that modern category recognizers are essentially quite dumb; so a `swimmer` recognizer looks mainly for water texture, and the `bomber␣plane` recognizer contains some tuning for "C" shapes corresponding to the airplane nose, and perhaps the "V" shapes at the wing and tail. Even if these recognizers are perhaps overspecialized for recognition of their nominal category, they can still provide useful building blocks to the learning algorithm that learns to recognize the novel class `duck`. Table 1 lists some highly-weighted classemes used to describe an arbitrarily selected subset of the Caltech256 categories. Each row of the table may be viewed as expressing the category as a weighted sum of building blocks; however the true building blocks are not the classeme labels that we can see, but their underlying dumb components, which we cannot. To complete the duck example, it is a combination of `body␣of␣water`, `bomber␣plane`, `swimmer`, as well as `walking` and `straight`. To gain an intuition as to what these categories actually represent, Figure 1 shows the training sets for the latter two. Examining the training images, we suggest that `walking` may represent "inverted V outdoors" and `straight` might correspond to "clutter and faces".

2 Background

Before describing the details of the system, and experimental investigations, we shall briefly summarize related literature.

The closest existing approach is probably image representation via *attributes* [11, 19]. Here object categories are described by a set of boolean attributes, such as "has beak", "no tail", "near water". Classifiers for these attributes are built by acquiring labels using Amazon's Mechanical Turk. In contrast, classemes are not designed to

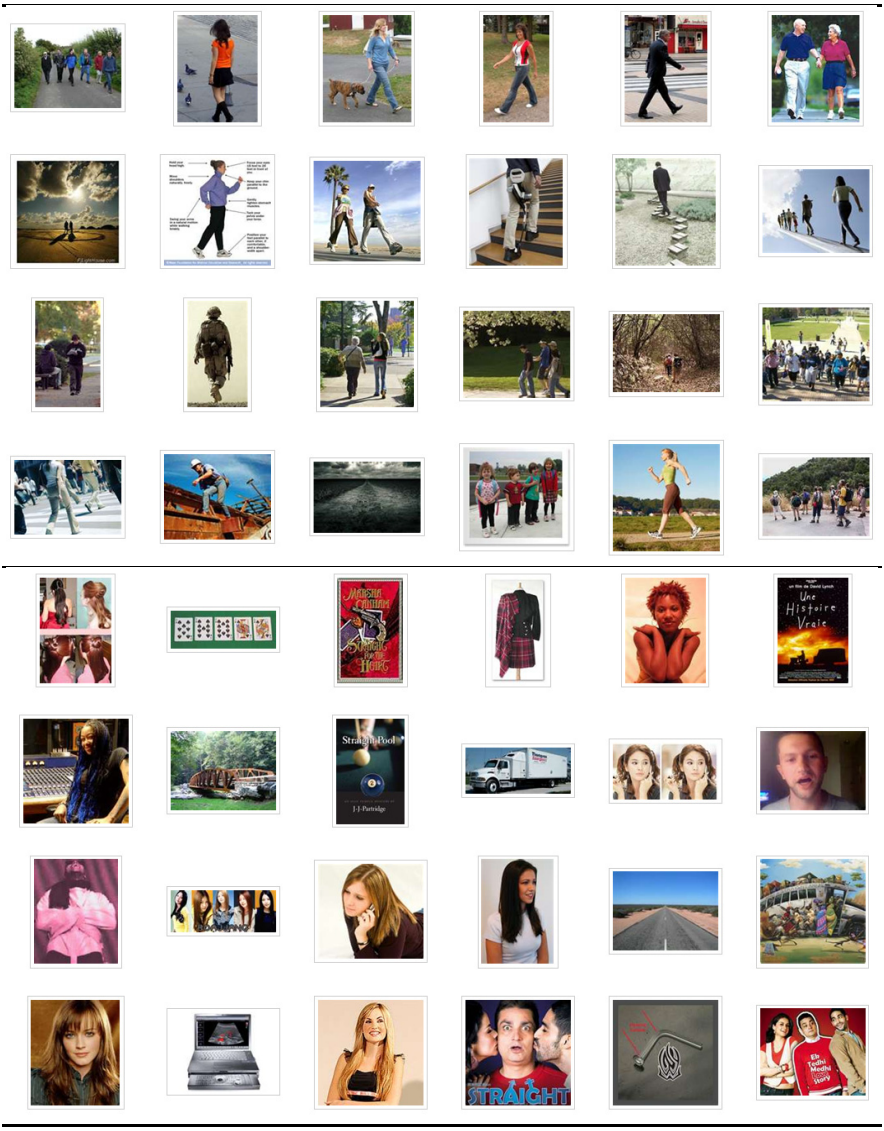

Fig. 1 Classeme training images. A subset of the training images for two of the 2659 classemes: `walking`, and `straight`. The top 150 training images are downloaded from Bing image search with no filtering or reranking. As discussed in the text, we do not require classeme categories to have a semantic relationship with the novel class; but to contain some building blocks useful for classification.

have specific semantic meanings, but rather to capture intersections of properties. Furthermore, they are trained using data directly obtained from web image search, without human cleanup. In addition, most prior attribute-based methods have relied on a "zero-shot" learning approach: instead of *learning* a classifier for a novel category from training examples, a user designs the classifier by listing attributes, limiting such systems to categories for which humans can easily extract attributes, and increasing the workload on the user even for such categories. A related idea is the representation of images in terms of distances to basis classes, which has been previously investigated as a way to define image similarities [42], to perform video search [15], or to enable natural scene recognition and retrieval [41].

The approach considered here is also evocative of Malisiewicz and Efros's "Recognition by Association" [23], in which object classes are represented by sets of object *instances* to which they are associated. In contrast, classemes represent object classes as a combination of other object *classes* to which they are related. This change of viewpoint enables the use of powerful kernel-based classifiers.

Because classemes represent images by a (relatively) low-dimensional feature vector, the approach is related to dimensionality reduction techniques and methods to learn compact codes for images [43, 36, 33, 31, 9]. These data-driven techniques find low-dimensional, typically nonlinear, projections of a large feature vector representing each image, such that the low-dimensional vectors are an effective proxy for the original. These techniques can achieve tremendous compressions of the image (for example to 64 bits [43]), but are of course correspondingly lossy, and have not been shown to be able to retain category-level information.

It is also useful to make a comparison to existing categorization systems in terms of how far they meet the requirements we have set out. In the discussion below, let N be the size of the test set (i.e. the image database, which may in principle be very large). Let n be the number of images in the training set, typically in the range $5-100$ per class. Let d be the dimensionality of the representation stored for each image. For example, if a histogram of visual words is stored, d is the minimum of the number of words detected per image and the vocabulary size. For a GIST descriptor [28], d is of the order of 1000. For multiple-kernel techniques [13], d might be of the order of $20,000$. For the system in this paper, d can be as low as 1500, while still leveraging all the descriptors used in the multiple-kernel technique. Note that although we shall later be specific about the number of bits per element of d, this is not required for the current discussion.

Boiman *et al.* [4] shows one of the most intriguing results on the Caltech256 benchmark: a nearest-neighbour-like classifier on low-level feature descriptors produces excellent performance, especially with small training sets. Its training cost is effectively zero: assemble a bag of descriptors from the supplied training images (although one might consider building a kd-tree or other spatial data structure to represent large training sets). However, the test-time algorithm requires that each descriptor in the *test* image be compared to the bag of descriptors representing the class, which has complexity $O(nd)$. It may be possible to build a kd-tree for the test set, and reverse the nearest-neighbor lookups, but the metric is quite asymmetric, so it is not at all clear that this will preserve the properties of the method.

For the multilple-kernel system of Gehler and Nowozin [13] the complexity is again $O(nd)$, but with large d, and a relatively large constant compared to the nearest-neighbor approach.

Another class of related techniques is the use of classifier combination other than multiple-kernel approaches. Zehnder *et al.* [44] build a classifier cascade which encourages feature sharing, but again requires the set of classes to be predefined, as is true for Griffin and Perona [14] and Torralba *et al.* [37]. Heitz *et al.* [16] propose to learn a general cascade similar to classemes (although with a different goal). However, the classeme approach simplifies training by pre-training the first layer, and simplifies testing by successfully working with simple top-layer classifiers.

3 Method Overview

The approach is now described precisely, but briefly, with more details supplied in §4. There are two distinct stages: once-only classeme learning; followed by any number of object-category-related learning tasks. Note that there are distinct training sets in each of the two stages.

3.1 Classeme Learning

A set of C category labels is drawn from an appropriate term list. For each category $c \in \{1..C\}$, a set of training images is gathered by issuing a query on the category label to an image search engine.

A one-versus-all classifier ϕ_c is trained for each category. The classifier output is real-valued, and is such that $\phi_c(\mathbf{x}) > \phi_c(\mathbf{x}')$ implies that $\mathbf{x}$ is more similar to class c than $\mathbf{x}'$ is. Given an image $\mathbf{x}$, then, the feature vector (descriptor) used to represent $\mathbf{x}$ is the *classeme vector* $\mathbf{f}(\mathbf{x}) = [\phi_1(\mathbf{x}), \ldots, \phi_C(\mathbf{x})]$.

Given the classeme vectors for all training images, it may be desired to perform some feature selection on the descriptors. We shall assume this has been done in the sequel, and simply write the classeme vector in terms of a reduced dimensionality $d \leq C$, so $\mathbf{f}(\mathbf{x}) = [\phi_1(\mathbf{x}), \ldots, \phi_d(\mathbf{x})]$. Where d is not specified it may be assumed that $d = C$.

Given the parameters of the ϕ_c, the training examples used to create the classemes may be discarded. We denote by Φ the set of functions $\{\phi_c\}_{c=1}^{d}$, which encapsulates the output of the classeme learning, and properly we shall write $\mathbf{f}(\mathbf{x}) = \mathbf{f}(\mathbf{x}; \Phi)$.

3.2 Using the Classemes

Given Φ, the rest of the approach is conventional. A typical situation might be that a new object category, or set of categories, is defined by a set of training images (note again that this is a new set of training images, unrelated to those used to build Φ). The training images are converted to classeme vectors, and then any classifier can be trained taking the classeme vectors as input. As shown in experiments, the features

are sufficiently powerful that simple linear classifiers applied to the classemes can give accuracies commensurate with much more expensive classifiers applied to the low-level image features. Useful candidate classifiers might be those which make a sparse linear combination of input features, so that the test cost is a small fraction of d per image; or predicate-based classifiers so that test images with nonzero score can be retrieved rapidly using inverted files [27, 38], achieving test complexity sublinear in N, the size of the test set.

4 Further Details

Several details are now expanded.

4.1 Selecting Category Labels

The set of category labels used to build the classemes should consist primarily of visual concepts. This will include concrete nouns, but may also include more abstract concepts such as "person working". The category labels should be chosen to be representative of the type of applications in which one plans to use the descriptors. As the focus of this study is general-category recognition, here we consider concepts selected from the Large Scale Concept Ontology for Multimedia (LSCOM) [26]. The LSCOM categories were developed specifically for multimedia annotation and retrieval, and have been used in the TRECVID video retrieval series. This ontology includes concepts selected to be useful, observable and feasible for automatic detection, and as such are likely to form a good basis for image retrieval and object recognition tasks. The LSCOM CYC ontology dated 2006-06-30 [22] was selected as the reference data set of concepts. From the initial 2832 unique concepts, the following categories were removed: 97 classes denoting abstract groups of other categories (marked in angle brackets in [22]); plural categories that also occurred as singulars; some people-related categories which were effectively near-duplicates. A total of $C = 2659$ categories were preserved by this filtering: the final list of concepts is available in [40]. Some examples have already been seen in table 1. This filtering was intentionally conservative in removing categories because, as discussed in the introduction, it is not easy to predict *a priori* what categories will be useful.

4.2 Gathering Category Training Data

For each category label, a set of training images was gathered by taking the top 150 images from the `bing.com` image search engine. For a general application these examples would not need to be manually filtered in any way, but in order to perform fair comparisons against the Caltech image database, near duplicates of images in that database were removed by a human-supervised process. Conversely, we did not remove overlap between the classeme *terms* and the Caltech categories

(28 categories overlap, see data on [40]), as a general-purpose system can expect to see overlap on a small number of queries. We also ran a test, not reported here, where classemes overlapping with Caltech256 labels were removed; the resulting performance was essentially unchanged.

4.3 Learning Classifiers ϕ_c

The classification model used for the $\phi_c(\cdot)$ is the LP-β kernel combiner of Gehler and Nowozin [13]. While they used 39 kernels, the experiments presented in this chapter are based on a set of 13 kernels. The kernels are defined in terms of the χ^2 distance between feature vectors as follows: $k(\mathbf{x},\mathbf{x}') = \exp(-\chi^2(\mathbf{x},\mathbf{x}')/\gamma)$, where γ is a hyper-parameter set as in [13] to be the average of the χ^2 distances in the training set. The following 13 feature types were used:

- *Kernel 1: Color GIST, d_1 = 960.* The GIST descriptor [28] is applied to color images. The images were resized to 32×32 (aspect ratio is not maintained), and then orientation histograms were computed on a 4×4 grid. Three scales were used with the number of orientations per scale being $8, 8, 4$.
- *Kernels 2-5: Pyramid of Histograms of Oriented Gradients, $d_{2..5} = 1700$.* The PHOG descriptor [7] is computed using 20 bins at four spatial pyramid scales.
- *Kernels 6-9: PHOG (2π unwrapped), $d_{6..9} = 3400$.* These features are obtained by using unoriented gradients quantized into 40 bins at four spatial pyramid scales.
- *Kernels 10-12: Pyramid self-similarity, $d_{10..12} = 6300$.* The Shechtman and Irani self-similarity descriptor [34] was computed as described by Bosch [5]. This gives a 30-dimensional descriptor at every 5th pixel. We quantized these descriptors into 5000 clusters using k-means, and a pyramid histogram was recorded with three spatial pyramid levels.
- *Kernel 13: Bag of words. $d_{13} = 5000$* SIFT descriptors [21] were computed at interest points detected with the Hessian-Affine detector [25]. These descriptors were then quantized using a vocabulary of size 5000, and accumulated in a sparse histogram.

A binary LP-β classifier was trained for each classeme, using a setup following the one described in section 7 of [13] in terms of kernel functions, kernel parameters, values of ν and number of cross validations. The only difference is that the objective of their equation (4) was modified in order to handle the uneven training set sizes. We used $N_+ = 150$ images as positive examples, and one image chosen at random from each of the other training sets as negative examples, so $N_- = C - 1$. The objective was modified by scaling the positive entries in the cost vector by (νN_+) and the negative entries by (νN_-). The cross-validation yields a per-class validation score which is used for feature selection.

4.4 Feature Selection

In order to perform feature selection on the classeme vectors $\mathbf{f}$, the classemes were first sorted in increasing order of cross-validation error. Given a desired feature dimensionality, d, the reduced classeme vector was obtained by selecting the first d components $\mathbf{f}(\mathbf{x}) = [\phi_1(\mathbf{x}), \ldots, \phi_d(\mathbf{x})]$. Again in situations where d is not specified it may be assumed that $d = C$

4.5 Classeme Quantization

For a practical system, the classeme vectors should not be stored in double precision, but instead an explicit quantization of the values should be used. This may be achieved by a simple quantization, or by defining binary "decision stumps" or predicates. Quantization can be performed either at novel-category learning time (i.e. on the novel training set) or at classeme-learning time. For 1-bit quantization, simple thresholding at 0 was used. For higher quantization numbers, the following "histogram-equalized" quantization was used. Given a training set of classeme vectors $\{\mathbf{f}_i\}_{i=1}^n$, write $\mathbf{f}_i = [\phi_{ik}]_{k=1}^d$. Write the rows of the matrix $[\mathbf{f}_1, \ldots, \mathbf{f}_n]$ as $\mathbf{r}_k = [\phi_{ik}]_{i=1}^n$. To quantize to Q levels, quantization centres z_{iq} are chosen as follows: $\mathbf{r}'_k = \text{sort}(\mathbf{r}_k)$, defining a matrix ϕ'_{ik}. Then make the set $Z_k = \{\phi'_{\lceil nq/(Q+1)\rceil,k}\}_{q=1}^Q$, and each value ϕ_{ik} is replaced by the closest value in Z_k.

5 Experiments

Given the simplicity of the approach, the first question that naturally arises is how it compares to the state-of-the-art recognition approaches. Here we compare to the LP-β kernel combiner as this is the current front-runner. Note that the key metric here is performance drop with respect to LP-β with the same 13 kernels used by classemes. As the classeme classifiers introduce an extra step in the recognition pipeline, performance might be expected to suffer from a "triangle inequality": the raw kernel combiner can optimize kernels for the final classes to recognize, while the classifiers using classemes as representation are forced to use the kernels trained on the LSCOM classes. The experiments show that this does happen, but to a small enough extent that the classemes remain competitive with the state of the art, and are much better than the closest "efficient" system.

There are two main experiments. In the first, we wish to assess the representational power of classemes with respect to existing methods, so we use the standard Caltech256 accuracy measure, with multiclass classifiers trained on all classes. In the second, we want to test classemes in a framework closer to their intended use, so we train one-vs-all classifiers on each class separately, and then report precision on ranking a set of images including distractors from the other classes.

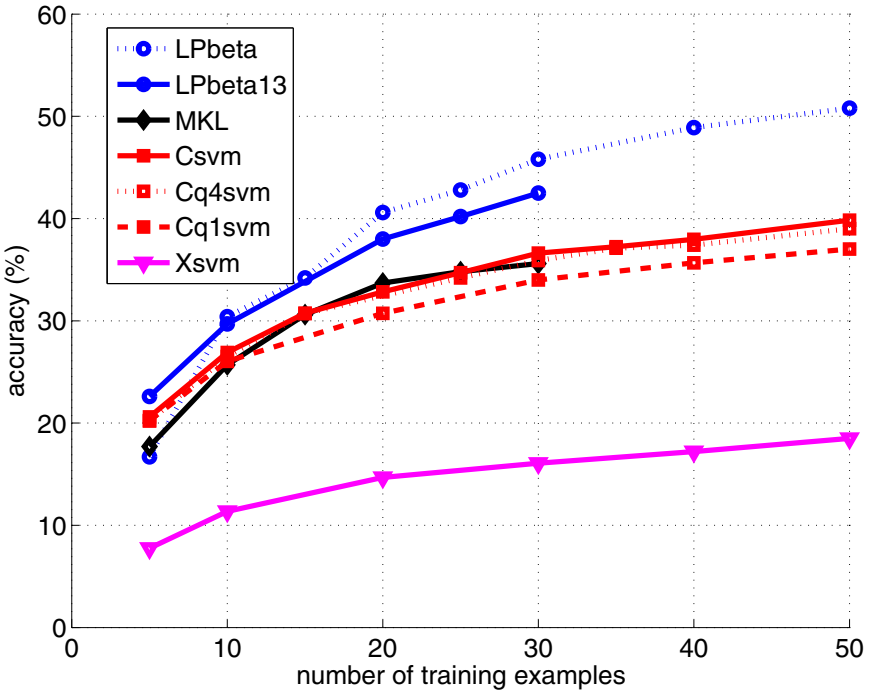

Fig. 2 Caltech256. A number of classifiers are compared on the Caltech256 dataset. **LPbeta** [13], **MKL**: Multiple Kernel learning [1], as implemented in [13], **LPbeta13**: LP-β on our low-level features (§4.3); **Xsvm** SVM trained on the concatenation of our low-level features. The classeme-based classifiers are: **Csvm**: SVM, floating point, $d = 1500$; **Cq4svm**: SVM, input quantized to 4 bits per channel (bpc), $d = 1500$; **Cq1svm**: SVM, input quantized to 1 bit, $d = 1500$. The key-result is this: on 30 training examples, and using the same underlying features, Csvm has 36% accuracy, and LPbeta13 has 42% accuracy, but the classeme-based system is orders of magnitude faster to train and test.

5.1 Experiment 1: Multiclass Classification

In this experiment we study the performance of classemes using the multiclass linear SVM of Joachims [18] as classification model, since this is an efficient classifier to train and test and thus it is well suited to our motivating problem. The SVM regularization parameter was set to be $\lambda = 3000$. All classeme-based results are presented for the case $d = 1500$, as using more than 1500 classemes was found to yield no further improvements.

Figures 2 shows the multi-class accuracy for different classifiers as a function of the number of training examples per class, using 25 test examples per category. It can be seen that the classeme-based SVM (Csvm) greatly outperforms an SVM directly trained on the same low-level features (Xsvm) and it matches the accuracy of the nonlinear classifier trained using multiple kernel learning [1]. Only LPbeta13

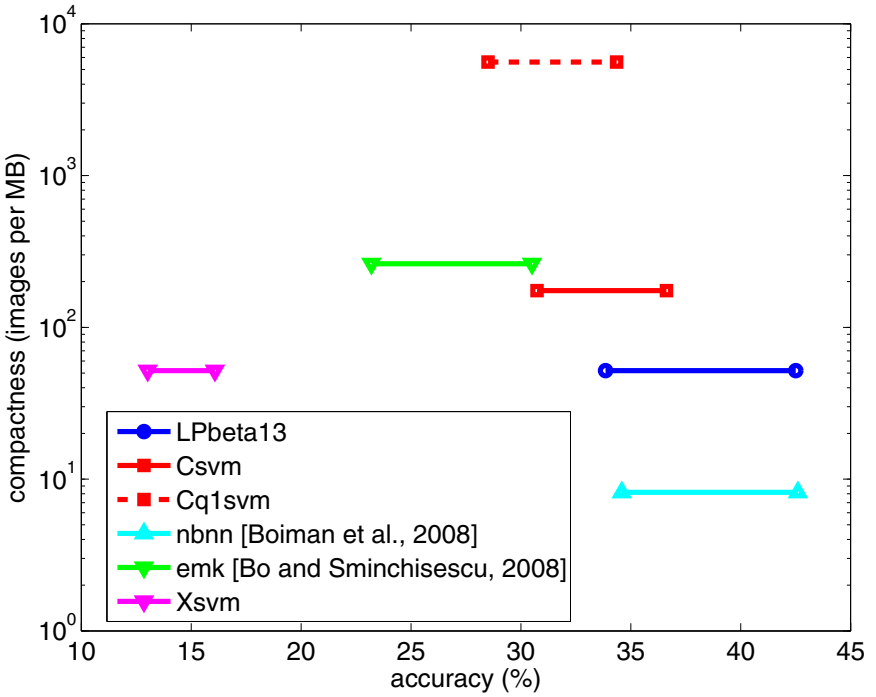

Fig. 3 Accuracy versus compactness of representation on Caltech-256. On both axes, higher is better. (Note logarithmic *y*-axis). The lines link performance at 15 and 30 training examples.

(the version of LP-β using the same low-level features exploited by classemes) provides higher accuracy. However the size of the representation is considerably reduced for classemes compared to LP-β: 2.5KB versus 23KB. Furthermore, the training and test times of our approach are considerably lower than LP-β: training the multiclass classifier Csvm with 5 examples for each Caltech class takes about 9 minutes on a AMD Opteron Processor 280 2.4GHz while the method of [13] requires more than 23 hours on the same machine; predicting the class of a text example takes 0.18ms with our model and 37ms with LP-β.

In addition, when moving from floating point classemes (Csvm) to a quantization of 4 bits per channel (Cq4svm) the change in accuracy is negligible. Accuracy drops by only 2–4 percentage points using a 1 bit per channel SVM (Cq1svm, $d = 1500$, 187.5 bytes per image). However, this representation increases the number of images that can be stored in an index by a factor of 100 over LP-β, which is especially significant for RAM-based indices.

Figure 3 shows accuracy versus compactness for different classification systems. In this plot we include also the performance of Naive Bayes Nearest Neighbor (nbnn) [4] and Efficient Match Kernel (EMK) [3]. It can be seen that classemes

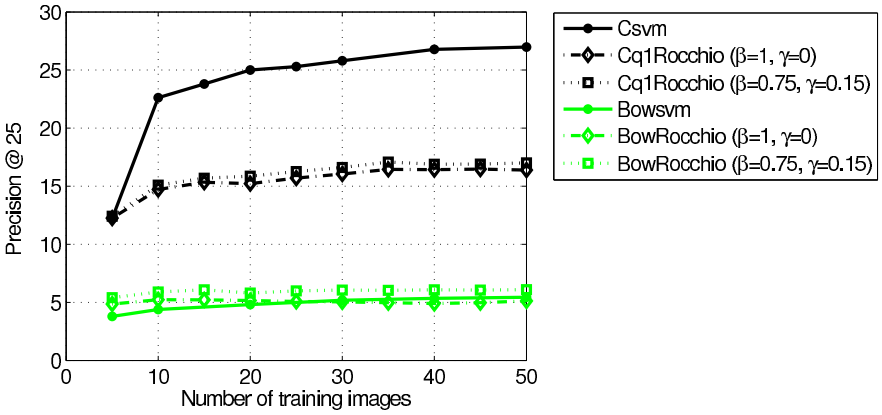

Fig. 4 Class retrieval in Caltech256. Percentage of the top 25 in a 6400-document set which match the query class. Random performance is 0.4%.

using 1 bit per channel provide a significant saving in terms of storage requirement compared to all other methods, while still yielding near state-of-the-art accuracy.

5.2 Experiment 2: Retrieval

The retrieval experiment attempts to gain insight into the behaviour of classemes in a class-retrieval task. A query against the database is specified by a set of training images taken from one category, and the retrieval task is to order the database by similarity to the query.

Evaluation on Caltech256. We start by studying performance on the Caltech256 data set. The test database is formed by sampling 25 images from each Caltech category. Success is measured as precision at 25: the proportion of the top 25 images which are in the same category as the query (training) set. The maximum score is 1, obtained if all the matching images are ranked above all the distractors. For this experiment, we compare classemes with bags of visual words (BOW), which are a popular model for efficient image retrieval. We use as BOW features the quantized SIFT descriptors of Kernel 13.

We consider two different retrieval methods. The first method is a linear SVM learned for each of the Caltech classes using the one-vs-the-rest strategy. We compare these classifiers to the Rocchio algorithm [24], which is a classic information retrieval technique for implementing relevance feedback. In order to use this method we represent each image as a document vector $\mathbf{d}(\mathbf{x})$. In the case of the BOW model, $\mathbf{d}(\mathbf{x})$ is the traditional *tf-idf*-weighted histogram of words. In the case of classemes instead, we define $\mathbf{d}(\mathbf{x})_i = [\phi_i(\mathbf{x}) > 0] \cdot \mathrm{idf}_i$, i.e. $\mathbf{d}(\mathbf{x})$ is computed by multiplying the binarized classemes by their inverted document frequencies. Given, a set of relevant training images D_r, and a set of non-relevant examples D_{nr}, Rocchio's algorithm

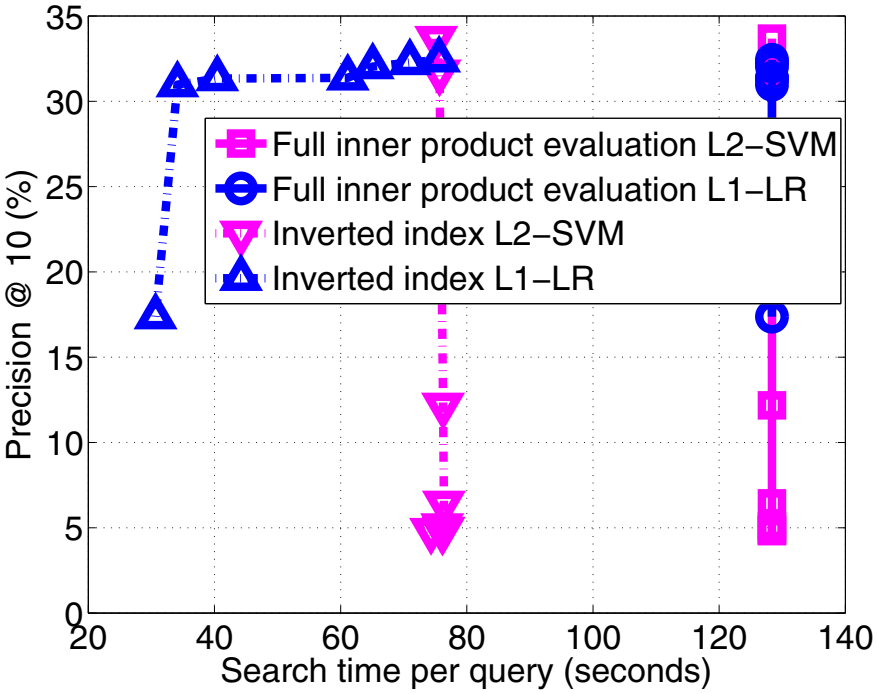

Fig. 5 Class-retrieval precision versus search time for the 10-million ImageNet database: *x*-axis is search time; *y*-axis shows percentage of true positives ranked in the top 10 (for each query class, the database contains $n^{-}_{test} = 9,671,611$ distractors and $n^{+}_{test} = 450$ true positives). The curve for each method is obtained by varying the hyperparameter in the learning objective of the classifier, thus producing different accuracy-speed tradeoffs (see details in the text).

computes the document query

$$\mathbf{q} = \beta \frac{1}{|D_r|} \sum_{\mathbf{x}_r \in D_r} \mathbf{d}(\mathbf{x}_r) - \gamma \frac{1}{|D_{nr}|} \sum_{\mathbf{x}_{nr} \in D_{nr}} \mathbf{d}(\mathbf{x}_{nr}) \tag{1}$$

where β and γ are scalar values. The algorithm then retrieves the database documents having highest *cosine similarity* with this query. In our experiment, we set D_r to be the training examples of the class to retrieve, and D_{nr} to be the remaining training images. We report results for two different settings: $(\beta, \gamma) = (0.75, 0.15)$, and $(\beta, \gamma) = (1, 0)$ corresponding to the case where only positive feedback is used.

Figure 4 shows that methods using classemes consistently outperform the algorithms based on traditional BOW features. Furthermore, SVM yields much better precision than Rocchio's algorithm when using classemes.

Evaluation on ImageNet (10M images). We now move on to present results on the large-scale ImageNet dataset [8], which includes about 10-million images representing over 15,000 categories (in this experiment we used 15,203 classes).

We randomly selected 400 categories as query classes. For each of these classes we capped the number of true positives in the database to be $n^+_{test} = 450$. The total number of distractors for each query is $n^-_{test} = 9,671,611$. Due to the size of the collection, we restrict our analysis to the binary version of classemes (1 bit per channel), using $d = 2659$.

We use this large-scale database to evaluate the speed-up achievable by implementing linear classification via inverted lists [24]. Inverted lists (also known as inverted indices) have been widely used for image search but predominantly for retrieval of near-duplicates or particular object-instances [35, 27, 30]. Instead here we adopt them to efficiently calculate the inner product between the weight vector learned at query-time and the binary classeme vector associated to each database image. This can be achieved by storing an inverted list for each classeme feature, enumerating the database images containing that particular classeme entry. The inverted lists allow the ranker to skip over classemes having value zero. A further speedup can be obtained by using a sparse classification model where the weight vector is constrained to have very few non-zero entries so that the evaluation cost will be a small fraction of the number of features (d). We use an ℓ_1-regularized logistic regression [10] (L1-LR) to test the advantages of a sparse classifier over the traditional ℓ_2-regularized SVM (L2-SVM).

For each query category we trained these two classification models using the one-vs-the-rest strategy, with a training set consisting of $n^+ = 10$ positive examples and $n^- = 15,202$ negative images obtained by sampling one training image for each of the negative classes. The results are summarized in figure 5. The x-axis shows average retrieval time per query, measured on a single-core computer with 16GB of RAM and an Intel Core i7-930 CPU @ 2.80GHz. The y-axis reports precision at 10 which measures the proportion of true positives in the top 10. The performance curve of each method was generated by varying the regularization hyperparameter λ in the learning objective of the classifier. While λ is traditionally viewed as controlling the bias-variance tradeoff, for the L1-LR classifier it can be interpreted as a parameter balancing generalization accuracy versus sparsity, and thus retrieval speed. It can be seen that inverted indices speed up considerably the retrieval, particularly in the case of L1-LR which tends to generate sparser weight vectors for which inverted indexing is especially advantageous: using this model ranking the entire 10-million dataset takes about 30 seconds, with an average precision@10 above 30%. As a reference, random retrieval would produce precision@10 roughly equivalent to 0.005%. Learning a L1-LR or an L2-SVM classifier for a query category in this experiment takes roughly 2 seconds.

We would like also to comment on the memory usage. Representing the database as a bit-map of all classemes would require a space of $(2659/8) \times N$ bytes for a database containing N images, which in this case amounts to about 3GB. The inverted list architecture requires more space. We represented the image IDs in inverted files using one byte per image: we achieve this by storing only ID displacements (which in our experiment happened to be always smaller than 255) between consecutive images in the list. Using this encoding, the total storage requirement for the 10M data set was roughly 9GB.

6 Discussion

In this chapter we have describe the learning of the classeme descriptor which is a representation intended to be useful for efficient high-level object recognition. By using the noisy training data from web image search in a novel way – to train "category-like" classifiers – the descriptor is essentially given access to knowledge about what humans consider "similar" when they search for images on the web (note that most search engines are considered to use "click-log" data to rank their image search results, so the results do reflect human preferences). The experiments have shown that this knowledge is effectively encoded in the classeme vector, and that this vector, even when quantized to below 200 bytes per image, gives competitive object category recognition performance.

A natural question is whether the weakly trained classemes actually do contain any semantic information, although we have emphasized that this is not the main motivation for their use.

We have focused here on object category recognition as characterized by the Caltech256 training data, which are adequate for clip-art search, but which will not be useful for, for example, home photo retrieval, or object indexing of surveillance footage. It should be straightforward to retrain the classemes on images such as the PASCAL VOC images, but a sliding-window approach would probably be required in order to achieve good performance.

Classemes were originally introduced in [39]. A further extension of this idea was presented in [2] where the classeme classifiers were trained jointly (as opposed to independently) by directly optimizing an objective measuring linear classification accuracy. A related approach is proposed by Li et al. [20] where the location-dependent output of object detectors evaluated on the image is used as a representation. The advantage of this descriptor is that it encodes spatial information; furthermore, object detectors are more robust to clutter and uninformative background than classifiers evaluated on the entire image. In [9] classemes were empirically shown to be useful also for low-level retrieval tasks such as finding images of the same scene as the query, particularly when used in conjunction with local-appearance descriptors [21, 29]. Also in [9], several binary encoding methods are presented to further compress the size of classemes while preserving their good retrieval properties. Some of these compression methods as well as a top-k ranking scheme are explored in [32] to further boost the efficiency of object-class retrieval in large databases using classemes.

Additional material including the list of classeme labels, the classeme training images, precomputed feature vectors for standard datasets, as well as software to extract this descriptor may be obtained from [40].

References

1. Bach, F.R., Lanckriet, G.R.G., Jordan, M.I.: Multiple kernel learning, conic duality, and the SMO algorithm. In: ICML (2004)
2. Bergamo, A., Torresani, L., Fitzgibbon, A.: Picodes: Learning a compact code for novel-category recognition. In: Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., Weinberger, K. (eds.) Advances in Neural Information Processing Systems 24, pp. 2088–2096 (2011)

3. Bo, L., Sminchisescu, C.: Efficient Match Kernel between Sets of Features for Visual Recognition. Adv. in Neural Inform. Proc. Systems (December 2009)
4. Boiman, O., Shechtman, E., Irani, M.: In defense of nearest-neighbor based image classification. In: Proc. Comp. Vision Pattern Recogn (CVPR) (2008)
5. Bosch, A.: Image classification using rois and multiple kernel learning (2010), http://eia.udg.es/~aboschr/Publicacions/bosch08a_preliminary.pdf
6. Chum, O., Philbin, J., Sivic, J., Isard, M., Zisserman, A.: Total recall: Automatic query expansion with a generative feature model for object retrieval. In: Intl. Conf. Computer Vision (2007)
7. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR (1), pp. 886–893 (2005)
8. Deng, J., Dong, W., Socher, R., Li, L., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR (2009)
9. Douze, M., Ramisa, A., Schmid, C.: Combining attributes and fisher vectors for efficient image retrieval. In: Proc. Comp. Vision Pattern Recogn, CVPR (2011)
10. Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., Lin, C.-J.: Liblinear: A library for large linear classification. J. of Machine Learning Research 9, 1871–1874 (2008)
11. Farhadi, A., Endres, I., Hoiem, D., Forsyth, D.: Describing objects by their attributes. In: Proc. Comp. Vision Pattern Recogn. (CVPR), pp. 1778–1785 (2009)
12. Fergus, R., Fei-Fei, L., Perona, P., Zisserman, A.: Learning object categories from google's image search. In: ICCV, pp. 1816–1823 (2005)
13. Gehler, P., Nowozin, S.: On feature combination for multiclass object classification. In: ICCV (2009)
14. Griffin, G., Perona, P.: Learning and using taxonomies for fast visual categorization. In: Proc. Comp. Vision Pattern Recogn. (CVPR) (2008)
15. Hauptmann, A.G., Yan, R., Lin, W.-H., Christel, M.G., Wactlar, H.D.: Can high-level concepts fill the semantic gap in video retrieval? a case study with broadcast news. IEEE Transactions on Multimedia 9(5), 958–966 (2007)
16. Heitz, G., Gould, S., Saxena, A., Koller, D.: Cascaded classification models: Combining models for holistic scene understanding. In: Advances in Neural Information Processing Systems (NIPS), pp. 641–648 (2008)
17. Jegou, H., Douze, M., Schmid, C.: Hamming embedding and weak geometric consistency for large scale image search. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part I. LNCS, vol. 5302, pp. 304–317. Springer, Heidelberg (2008)
18. Joachims, T.: An implementation of support vector machines (svms) in c (2002)
19. Lampert, C.H., Nickisch, H., Harmeling, S.: Learning to detect unseen object classes by between-class attribute transfer. In: Proc. Comp. Vision Pattern Recogn. (CVPR) (2009)
20. Li-Jia Li, E.P.X., Su, H., Fei-Fei, L.: Object bank: A high-level image representation for scene classification semantic feature sparsification. In: NIPS (2010)
21. Lowe, D.: Distinctive image features from scale-invariant keypoints. Intl. Jrnl. of Computer Vision 60(2), 91–110 (2004)
22. LSCOM (2006), http://lastlaugh.inf.cs.cmu.edu/lscom/ontology/LSCOM-20060630.txt, http://www.lscom.org/ontology/index.html (Cyc ontology dated June 30, 2006)
23. Malisiewicz, T., Efros, A.A.: Recognition by association via learning per-exemplar distances. In: Proc. Comp. Vision Pattern Recogn. (CVPR) (2008)
24. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge Univ. Press (2008)

25. Mikolajczyk, K., Schmid, C.: Scale and affine invariant interest point detectors. Intl. Jrnl. of Computer Vision 60(1), 63–86 (2004)
26. Naphade, M., Smith, J.R., Tesic, J., Chang, S.-F., Hsu, W., Kennedy, L., Hauptmann, A., Curtis, J.: Large-scale concept ontology for multimedia. IEEE MultiMedia 13(3), 86–91 (2006)
27. Nistér, D., Stewénius, H.: Scalable recognition with a vocabulary tree. In: Proc. Comp. Vision Pattern Recogn. (CVPR), pp. 2161–2168 (2006)
28. Oliva, A., Torralba, A.: Building the gist of a scene: The role of global image features in recognition. Visual Perception, Progress in Brain Research 155 (2006)
29. Perronnin, F., Sánchez, J., Mensink, T.: Improving the fisher kernel for large-scale image classification. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part IV. LNCS, vol. 6314, pp. 143–156. Springer, Heidelberg (2010)
30. 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)
31. Raginsky, M., Lazebnik, S.: Locality-sensitive binary codes from shift-invariant kernels. In: Advances in Neural Information Processing Systems (NIPS) (2010)
32. Rastegari, M., Fang, C., Torresani, L.: Scalable object-class retrieval with approximate and top-k ranking. In: ICCV, pp. 2659–2666 (2011)
33. Salakhutdinov, R., Hinton, G.: Semantic hashing. Int. J. Approx. Reasoning 50, 969–978 (2009)
34. Shechtman, E., Irani, M.: Matching local self-similarities across images and videos. In: Proc. Comp. Vision Pattern Recogn. (CVPR) (June 2007)
35. Sivic, J., Zisserman, A.: Video Google: A text retrieval approach to object matching in videos. In: ICCV (2003)
36. Torralba, A., Fergus, R., Weiss, Y.: Small codes and large image databases for recognition. In: Proc. Comp. Vision Pattern Recogn. (CVPR) (2008)
37. Torralba, A., Murphy, K.P., Freeman, W.T.: Sharing visual features for multiclass and multiview object detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 29(5), 854–869 (2007)
38. Torresani, L., Szummer, M., Fitzgibbon, A.: Learning query-dependent prefilters for scalable image retrieval. In: Proc. Comp. Vision Pattern Recogn. (CVPR), pp. 2615–2622 (2009)
39. Torresani, L., Szummer, M., Fitzgibbon, A.: Efficient object category recognition using classemes. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part I. LNCS, vol. 6311, pp. 776–789. Springer, Heidelberg (2010)
40. Torresani, L., Szummer, M., Fitzgibbon, A.: Efficient object category recognition using classemes, web page (2010), `http://www.cs.dartmouth.edu/~lorenzo/projects/classemes`
41. Vogel, J., Schiele, B.: Semantic modeling of natural scenes for content-based image retrieval. Intl. Jrnl. of Computer Vision 72(2), 133–157 (2007)
42. Wang, G., Hoiem, D., Forsyth, D.: Learning image similarity from flickr using stochastic intersection kernel machines. In: Intl. Conf. Computer Vision (2009)
43. Weiss, Y., Torralba, A., Fergus, R.: Spectral hashing. In: NIPS (2008)
44. Zehnder, P., Koller-Meier, E., Gool, L.V.: An efficient shared multi-class detection cascade. In: British Machine Vision Conf. (2008)