Efficient Object Category Recognition Using Classemes

Lorenzo Torresani[1], Martin Szummer[2], and Andrew Fitzgibbon[2]

[1] Darthmouth College, Hanover, NH, USA
www.cs.dartmouth.edu/~lorenzo
[2] Microsoft Research, Cambridge, United Kingdom
http://www.research.microsoft.com/~{szummer,awf}

Abstract. We introduce a new descriptor for images which allows the construction of efficient and compact classifiers with good accuracy on object category recognition. The descriptor is the output of a large number of weakly trained object category classifiers on the image. The trained categories are selected from an ontology of visual concepts, but the intention is not to encode an explicit decomposition of the scene. Rather, we accept that existing object category classifiers often encode not the category *per se* but ancillary image characteristics; and that these ancillary characteristics can combine to represent visual classes unrelated to the constituent categories' semantic meanings.

The advantage of this descriptor is that it allows object-category queries to be made against image databases using efficient classifiers (efficient at test time) such as linear support vector machines, and allows these queries to be for novel categories. Even when the representation is reduced to 200 bytes per image, classification accuracy on object category recognition is comparable with the state of the art (36% versus 42%), but at orders of magnitude lower computational cost.

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 paper identifies three requirements for such scaling, and proposes a new descriptor which satisfies them.

We suggest that interesting large-scale applications must recognize **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. Note that kernel-based classifiers, which represent the current state of the art, do not satisfy this requirement because the (kernelized) distance between each database image and (a subset of) the novel training images must be computed. Without the novel-category requirement, the problem is trivial—the search results can be precomputed by running the known category detector on each database image at ingestion time, and storing the results as inverted files.

K. Daniilidis, P. Maragos, N. Paragios (Eds.): ECCV 2010, Part I, LNCS 6311, pp. 776–789, 2010.

Table 1. Highly weighted classemes. Five classemes with the highest LP-β weights for the retrieval experiment, for a selection of Caltech 256 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

Large-scale recognition benefits from a **compact descriptor** for each image, for example allowing databases to be stored in memory rather than on disk. The descriptor we propose is 2 orders of magnitude more compact than the state of the art, at the cost of a small drop in accuracy. In particular, performance of the state of the art with 15 training examples is comparable to our most compact descriptor with 30 training examples.

The ideal descriptor also provides good results with **simple classifiers**, such as linear SVMs, decision trees, or tf-idf, as these can be implemented to run efficiently on large databases.

Although a number of systems satisfy these desiderata for object instance or place recognition [18,9] or for whole scene recognition [26], we argue that no existing system has addressed these requirements in the context of object category recognition.

The system we propose is a form of classifier combination, the components of the proposed descriptor are the outputs of a set of predefined category-specific classifiers applied to the image. The obvious (but only partially correct) intuition is that a novel category, say `duck`, will be expressed in terms of the outputs of base classifiers (which we call "classemes"), describing either objects similar to ducks, or objects seen in conjunction with ducks. Because these base classifier outputs provide a rich coding of the image, simple classifiers such as linear SVMs can approach state-of-the art accuracy, satisfying the requirements listed above. However, 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, we work on the assumption 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, and discuss how existing systems fit the requirements.

The closest existing approach is probably image representation via *attributes* [5,11]. 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, we do not design our attributes to have specific semantic meanings, but expect meaning to emerge from intersections of properties, and we obtain training data directly from web image search without human cleanup. Furthermore, 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.

Our approach is also evocative of Malisiewicz and Efros's "Recognition by Association" [14], in which object classes are represented by sets of object *instances* to which they are associated. In contrast, we represent object classes as a combination of other object *classes* to which they are related. This change of viewpoint allows us to use the powerful classifiers produced by recent advances in object category recognition.

Because we represent images by a (relatively) low-dimensional feature vector, our approach is related to dimensionality reduction techniques exemplified by *semantic hashing* [20,26]. 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 [26]), 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 systems which, while less related in form, represent the state of the art in object category recognition. The assessment is thus in terms of how far the existing systems meet the requirements we

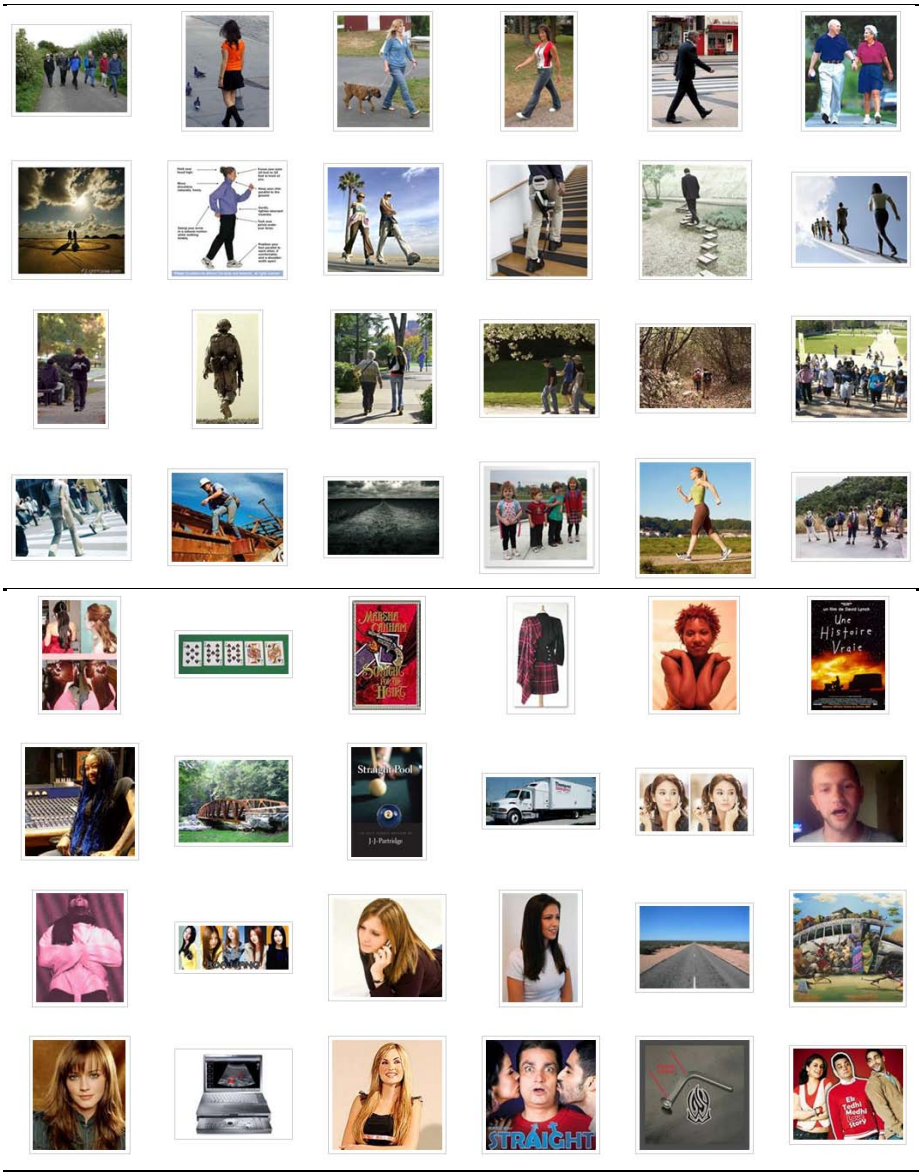

Fig. 1. Classeme training images. A subset of the training images for two of the 2659 classemes: `walking`, and `straight`. The top 150 training images are downloaded from Bing image search with no filtering or reranking. As discussed in the text, we do not require classeme categories to have a semantic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

have 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 [19], d is of the order of 1000. For multiple-kernel techniques [6], 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.* [2] shows one of the most intriguing results on the Caltech 256 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.

Gehler and Nowozin [6] represents the state of the art in classification accuracy on the Caltech 256 benchmarks, using a kernel combination classifier. However, training this classifier is expensive, and more importantly, test-time evaluation requires that several kernels be evaluated between each test image and several images of the training set. Note that these kernels cannot be precomputed, because in our problem the training set is different for every new query. Therefore 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.* [27] 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 [7] and Torralba *et al.* [23]. Heitz *et al.* [8] propose to learn a general cascade similar to ours (although with a different goal), but our approach simplifies training by pre-training the first layer, and simplifies test by successfully working with simple top-layer classifiers.

3 Method Overview

Our 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 $\phi_c(\mathbf{x}) > \phi_c(\mathbf{y})$ implies that $\mathbf{x}$ is more similar to class c than $\mathbf{y}$ 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 our 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 we show in experiments, the features are sufficiently powerful that simple and fast 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 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 [18,24], 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". Although we know of no general rules for category selection, the category labels should probably be chosen to be representative of the type of applications in which one plans to use the

descriptors. As we considered general-category image search as a target application domain, we selected concepts from the Large Scale Concept Ontology for Multimedia (LSCOM) [17]. 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. We took the LSCOM CYC ontology dated 2006-06-30 [13], which contains 2832 unique categories. We removed 97 categories denoting abstract groups of other categories (marked in angle brackets in [13]), and then removed plural categories that also occurred as singulars, and some people-related categories which were effectively near-duplicates, and arrived at $C = 2659$ categories. Some examples have already been seen in table 1. We were 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 supplementary data on [25]), as a general-purpose system can expect to see overlap on a small number of queries. However, we do include one test (CsvmN, figure 2) which shows no significant drop in performance by removing these terms.

4.3 Learning Classifiers ϕ_c

The learning algorithm used for the $\phi(\cdot)$ is the LP-β kernel combiner of Gehler and Nowozin [6]. They used 39 kernels, but we reduced this to 13 for experimentation. We employed kernels defined in terms of χ^2 distance between feature vectors, i.e. $k(x, x') = \exp(-\chi^2(x, x')/\gamma)$, using the following 13 feature types:

- *Kernel 1: Color GIST, $d_1 = 960$.* The GIST descriptor [19] 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 [4] 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 [21] was computed as described by Bosch [3]. This gives a 30-dimensional descriptor at every 5th pixel. We quantized these descriptors into 300 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 [12] were computed at interest points detected with the Hessian-Affine detector [16]. 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 [6] 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

We perform some simple dimensionality reduction of the classeme vectors $\mathbf{f}$ as follows. The classemes are sorted in increasing order of cross-validation error. Given a desired feature dimensionality, d, the reduced classeme vector is then simply 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 we just thresholded at 0. For higher numbers, we use the following "histogram-equalized" quantization. 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: $\mathbf{r}'_k = \mathrm{sort}(\mathbf{r}_k)$, defining a row-sorted 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 13 kernels, as this means that the base features are the same between LP-β and classemes.

As our 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 on the d_{LP} features directly, while the classeme classifiers are forced to go via the d classemes. We will 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 Caltech 256 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 Caltech class, and then report precision on ranking a set of images including distractors from the other classes.

5.1 Experiment 1: Multiclass Classification

To use classemes for multiclass classification, several strategies were implemented: multiclass SVMs [10], neural networks, decision forests [22] and a nearest-neighbour classifier. Comments on each of these follow. The variable T is the number of training examples per class. Figures 2 and 3 summarize the results.

Multiclass SVMs were trained using the SVMlight software [10], with regularization parameter $\lambda = 3000$, and $d = 1500$. The regularization parameter was determined by cross-validation for $T = 15$, and fixed for all subsequent tests.

Decision trees were trained using a standard information gain criterion. Because the small training set size limits the tree depth (depth 9 was found by cross-validation at $T = 15$), decision forests were found to require large forests (around 50 trees), and were not included in subsequent testing. Similarly, a nearest-neighbour classifier was tested and found to give low performance, so it is not included in these results, but complete results can be seen at [25].

A standard 1-hidden-layer network with tanh nonlinearities and a softmax over 256 outputs was trained. The number of hidden units ranged between 500 and 1000, chosen on a validation set, and training used an L_1 weight decay regularizer fixed at 0.1 and was stopped early at 40–80 iterations.

For each class, the number of training images was varied, and 25 test images were used. Performance is quoted as the mean of average accuracy per class as in [6], and plotted in figure 2. It can be seen that the classeme-based neural network (Cnet) and SVM classifiers (Csvm) beat all but LP-β and NBNN. However the size of the representation is considerably reduced for classemes compared to LP-β and NBNN: $2.5K$ for classemes versus $23K$ and $128K$ respectively. 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

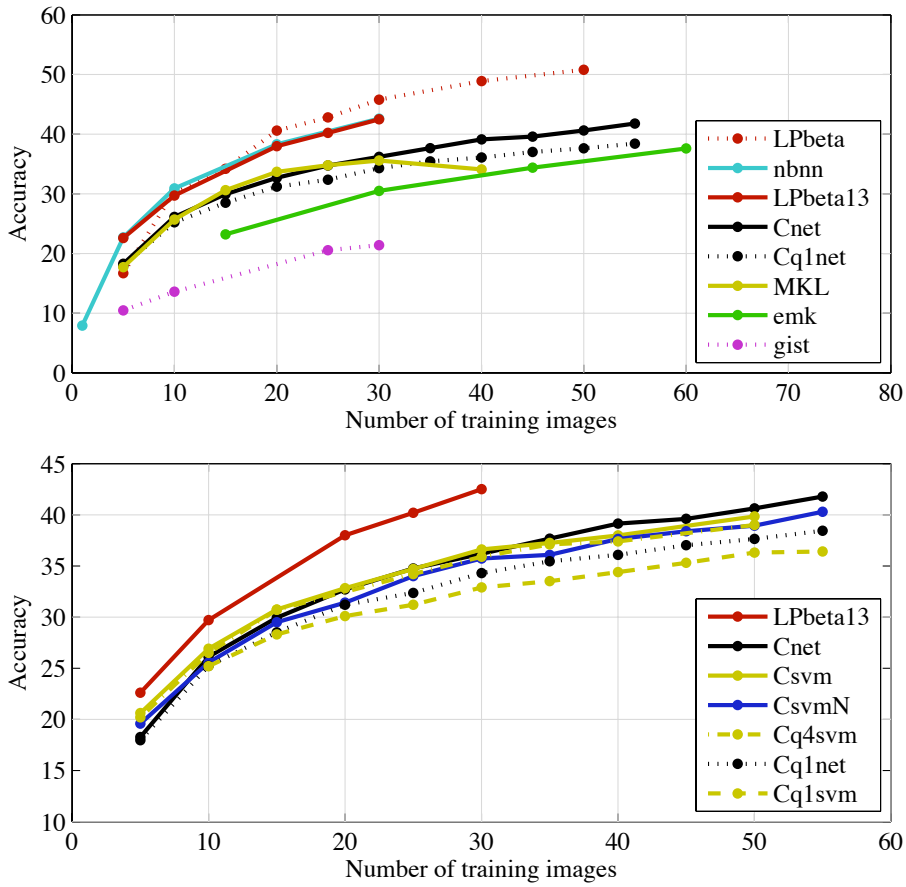

Fig. 2. Caltech 256. A number of classifiers are compared on the Caltech 256 dataset. The key claim is this: on 30 training examples, and using the same underlying features, Cnet1q1 and Csvm have 36% accuracy, and LPbeta13 has 42% accuracy, but the classeme-based systems are orders of magnitude faster to train and test.
(**Top**): Classeme neural net compared to results in the literature: **LPbeta** [6], **NBNN**: Naive Bayes Nearest Neighbor [2]; **MKL**: Multiple Kernel learning, as implemented in [6]; **EMK**: Efficient Match Kernel [1]. In addition we add our implementations of: **LPbeta13** (LP-β on our base features §4.3); **GIST**: One-vs-all SVM on GIST feature alone; **Cnet**: Neural network trained on floating point classeme vector; **Cq1net**: Neural network on 1 bit-per-channel (1bpc) classeme vector.
(**Bottom**): Comparison of classeme-based classifiers. (Except LPbeta13, included for reference). **Csvm**: SVM, floating point, $d = 1500$; **CsvmN**: SVM, floating point, Caltech terms removed from training (§4.2); **Cq4svm**: SVM, input quantized to 4 bits per channel (bpc), $d = 1500$; **Cq1svm**: SVM, input quantized to 1 bit, $d = 1500$.

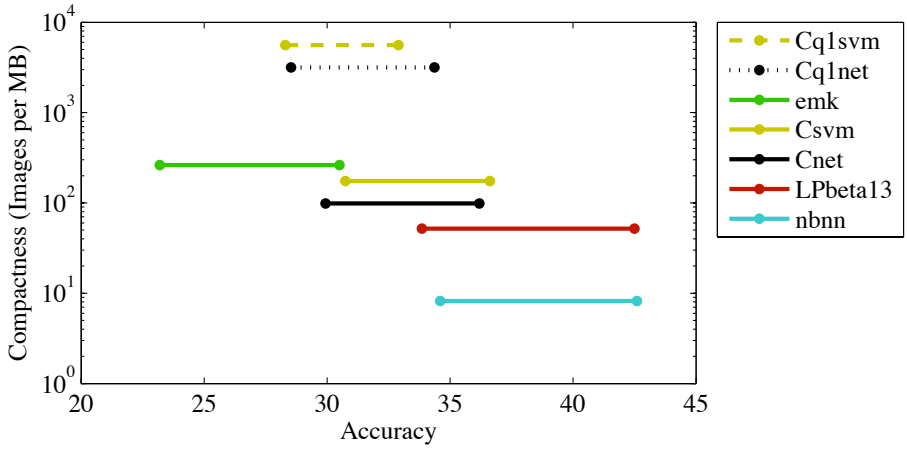

Fig. 3. Accuracy versus compactness of representation on Caltech-256. On both axes, higher is better. (Note logarithmic y-axis). The lines link performance at 15 and 30 training examples.

the method of [6] requires more than 23 hours on the same machine; predicting the class of a test example takes 0.18ms with our model and 37ms with LP-β.

Furthermore, 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 1–2 percentage points using a 1 bit per channel neural network (Cq1net, 312 bytes per image), and 1–2 more using a 1bpc SVM (Cq1svm, $d = 1500$, 187.5 bytes per image). This storage requirement 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.

Also plotted for reference is the accuracy of GIST as a single feature, being an important contributor to LP-β's kernel pool. We note that the GIST vector at 960 bytes is already much longer than the 187.5 bytes of Cq1svm while being much less accurate.

5.2 Experiment 2: Retrieval

The retrieval experiment attempts to gain insight into the behaviour of classemes in a retrieval task. The test database is a concatenation of 25 images from each Caltech category. 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. Success is measured as precision at 25: the proportion of the top 25 images which are in the same category as the query 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.

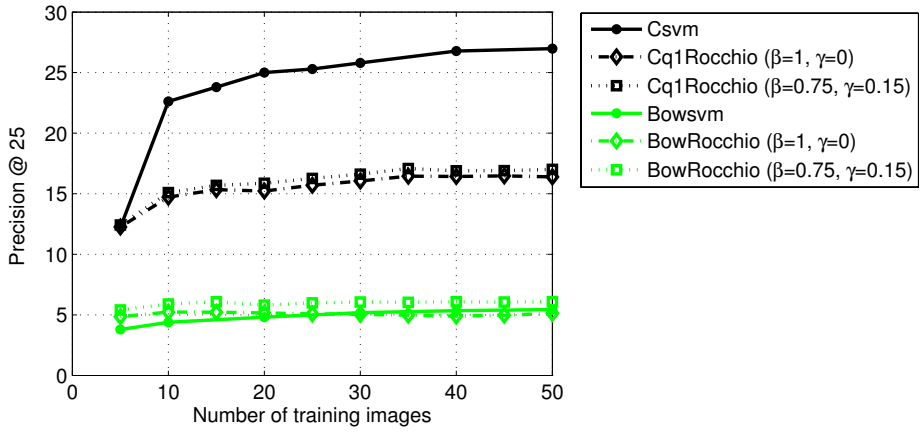

Fig. 4. Retrieval. Percentage of the top 25 in a 6400-document set which match the query class. Random performance is 0.4%.

We consider two different retrieval methods. The first method is a linear SVM learned for each of the Caltech classes using the one-vs-all strategy. We compare these classifiers to the Rocchio algorithm [15], 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 $\mathbf{d}(\mathbf{x})_i = [\phi_i(\mathbf{x}) > 0] \cdot \text{idf}_i$, 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 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. Note that these linear classifiers can be evaluated very efficiently even on large data sets; furthermore, they can also be trained efficiently and thus used in applications requiring fast query-time learning: for example, the average time required to learn a one-vs-all SVM using classemes is 674 ms when using 5 training examples from each Caltech class.

6 Discussion

We have introduced a new descriptor for images which is intended to be useful for 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). We 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.

An important question is whether the weakly trained classemes actually do contain any semantic information. We have emphasized throughout the paper that this is not the main motivation for their use, and we do so again here. It may be that one might view the classemes as a form of highly nonlinear random projection, and it is interesting future work to see if something close to random splits will yield equivalent performance.

We have focused here on object category recognition as characterized by the Caltech 256 training data, which we consider 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.

Several avenues remain open to improve these results. Our feature selection from among the 2659 raw features is currently very rudimentary, and it may be helpful to apply a sparse classifier. The various hashing techniques can immediately be applied to our descriptor, and might result in considerable reductions in storage and computational cost.

Additional material including the list of classemes, the retrieved training images, and precomputed classeme vectors for standard datasets, may be obtained from [25].

References

1. Bo, L., Sminchisescu, C.: Efficient Match Kernel between Sets of Features for Visual Recognition. In: Adv. in Neural Inform. Proc. Systems (December 2009)
2. Boiman, O., Shechtman, E., Irani, M.: In defense of nearest-neighbor based image classification. In: Proc. Comp. Vision Pattern Recogn., CVPR (2008)
3. Bosch, A.: Image classification using rois and multiple kernel learning (2010), `http://eia.udg.es/~aboschr/Publicacions/bosch08a_preliminary.pdf`
4. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR, vol. 1, pp. 886–893 (2005)
5. Farhadi, A., Endres, I., Hoiem, D., Forsyth, D.: Describing objects by their attributes. In: Proc. Comp. Vision Pattern Recogn. (CVPR), pp. 1778–1785 (2009)
6. Gehler, P.V., Nowozin, S.: On feature combination for multiclass object classification. In: Intl. Conf. Computer Vision (2009)

7. Griffin, G., Perona, P.: Learning and using taxonomies for fast visual categorization. In: Proc. Comp. Vision Pattern Recogn., CVPR (2008)
8. Heitz, G., Gould, S., Saxena, A., Koller, D.: Cascaded classification models: Combining models for holistic scene understanding. In: Adv. in Neural Inform. Proc. Systems (NIPS), pp. 641–648 (2008)
9. Jegou, H., Douze, M., Schmid, C.: Hamming embedding and weak geometric consistency for large scale image search. In: European Conf. Comp. Vision (October 2008)
10. Joachims, T.: An implementation of support vector machines (svms) in c (2002)
11. 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)
12. Lowe, D.: Distinctive image features from scale-invariant keypoints. Intl. Jrnl. of Computer Vision 60(2), 91–110 (2004)
13. LSCOM: Cyc ontology dated (2006-06-30), `http://lastlaugh.inf.cs.cmu.edu/lscom/ontology/LSCOM-20060630.txt`, `http://www.lscom.org/ontology/index.html`
14. Malisiewicz, T., Efros, A.A.: Recognition by association via learning per-exemplar distances. In: Proc. Comp. Vision Pattern Recogn., CVPR (2008)
15. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, Cambridge (2008)
16. Mikolajczyk, K., Schmid, C.: Scale and affine invariant interest point detectors. Intl. Jrnl. of Computer Vision 60(1), 63–86 (2004)
17. 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)
18. Nistér, D., Stewénius, H.: Scalable recognition with a vocabulary tree. In: Proc. Comp. Vision Pattern Recogn. (CVPR), pp. 2161–2168 (2006)
19. 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)
20. Salakhutdinov, R., Hinton, G.: Semantic hashing. In: SIGIR Workshop on Information Retrieval and Applications of Graphical Models (2007)
21. Shechtman, E., Irani, M.: Matching local self-similarities across images and videos. In: Proc. Comp. Vision Pattern Recogn. CVPR (June 2007)
22. Shotton, J., Johnson, M., Cipolla, R.: Semantic texton forests for image categorization and segmentation. In: Proc. Comp. Vision Pattern Recogn., CVPR (2008)
23. 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)
24. 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)
25. Torresani, L., Szummer, M., Fitzgibbon, A.: Efficient object category recognition using classemes (2010), `http://www.cs.dartmouth.edu/~lorenzo/projects/classemes`
26. Weiss, Y., Torralba, A.B., Fergus, R.: Spectral hashing. In: Adv. in Neural Inform. Proc. Systems (NIPS), pp. 1753–1760 (2008)
27. Zehnder, P., Koller-Meier, E., Gool, L.V.: An efficient shared multi-class detection cascade. In: British Machine Vision Conf. (2008)