PiCoDes: Learning a Compact Code for Novel-Category Recognition

Alessandro Bergamo **Lorenzo Torresani**
Computer Science Department - Dartmouth College
Hanover, NH 03755, U.S.A.
{aleb, lorenzo}@cs.dartmouth.edu

Andrew Fitzgibbon
Microsoft Research
Cambridge, United Kingdom
awf@microsoft.com

Abstract

We introduce PiCoDes: a very compact image descriptor which nevertheless allows high performance on object category recognition. In particular, we address novel-category recognition: the task of defining indexing structures and image representations which enable a large collection of images to be searched for an object category, where the training images defining the category are supplied at query time. We explicitly learn descriptors of a target length (from as small as 16 bytes per image) which have good object-recognition performance. In contrast to previous work in the domain of object recognition, we do not choose an arbitrary intermediate representation, but explicitly learn short codes. In contrast to previous approaches to learn compact codes, we optimize explicitly for (an upper bound on) classification performance. Optimization directly for binary features is difficult and nonconvex, but we present an alternation scheme and convex upper bound which demonstrate good performance in practice. When compared with existing proposals for compact category recognition, we demonstrate several operating points outside the current compactness/accuracy envelope.

1 Introduction

In this work we consider the problem of efficient object-class recognition in large image collections. We are specifically interested in 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 from a large database images belonging to this class. This application scenario 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 rather than on disk for additional efficiency.

Traditional object categorization methods do not meet these requirements as they employ high-dimensional descriptors and they typically use non-linear kernels which render them computationally expensive to train and test. For example, the LP-β multiple kernel combiner [10] achieves state-of-the-art accuracy on several categorization benchmarks but it requires over 23 Kbytes to represent each image and it uses 39 feature-specific nonlinear kernels. This recognition model is infeasible for our application because it would require costly *query-time* kernel evaluations for each image in the database since the training set varies with every new query and thus pre-calculation of kernel distances is not possible.

We propose to address these storage and efficiency requirements by learning a compact binary image representation, called PiCoDes[1], optimized to yield good categorization accuracy with efficient

[1]Which we think of as "Picture Codes" or "Pico-Descriptors", or (with Greek pronunciation) π-codes

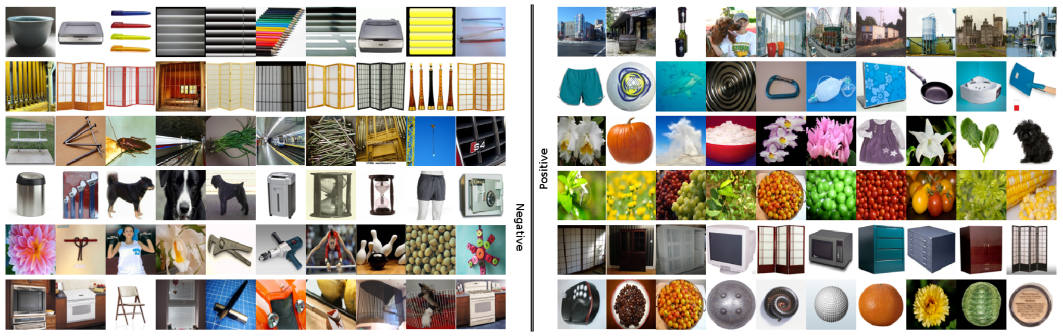

Figure 1: **Visualization of PiCoDes.** The 128-bit PiCoDe (whose accuracy on Caltech256 is displayed in figure 3) is applied to the test data of ILSVRC2010. Six of the 128 bits are illustrated as follows: for bit c, all images are sorted by non-binarized classifier outputs $\boldsymbol{a}_c^\top \boldsymbol{x}$ and the 10 smallest and largest are presented on each row. Note that $\boldsymbol{a}_c$ is defined only up to sign, so the patterns to which the bits are specialized may appear in either the "positive" or "negative" columns.

(e.g. linear) classifiers. The binary entries in our image descriptor are thresholded nonlinear projections of low-level visual features extracted from the image, such as descriptors encoding texture or the appearance of local image patches. Each non-linear projection can be viewed as implementing a nonlinear classifier using multiple kernels. The intuition is that we can then use these pre-learned multiple kernel combiners as a **classification basis** to define recognition models for arbitrary novel categories: the final classifier for a novel class is obtained by linearly combining the binary outputs of the basis classifiers, which we can pre-compute for every image in the database, thus enabling efficient novel object-class recognition even in large datasets.

The search for compact codes for images has been the subject of much recent work, which we loosely divide into "designed" and "learned" codes. In the former category we include min-hash [5], VLAD [12], and attributes [9, 15, 14] which are fully-supervised classifiers trained to recognize certain visual properties in the image. 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 [27], to perform video search [11], or to enable natural scene recognition and retrieval [26]. Torresani et al. [24] define a compact image code as a bitvector, the entries of which are the outputs of a large set of weakly-trained basis classifiers ("classemes") evaluated on the image. Simple linear classifiers trained on classeme vectors produce near state-of-the-art categorization accuracy. Li et al. [16] use the localized outputs of object detectors as an image representation. The advantage of this representation is that it encodes spatial information; furthermore, object detectors are more robust to clutter and uninformative background than classifiers evaluated on the entire image. These prior methods work under the assumption that an "overcomplete" representation for classification can be obtained by pre-learning classifiers for a large number of basis classes, some of which will be related to those encountered at test-time. Such high-dimensional representations are then compressed down using quantization, dimensionality reduction or feature selection methods.

The second strand of related work is the *learning* of compact codes for images [28, 23, 21, 19, 7] where binary image codes are learned such that the Hamming distance between codewords approximates a kernelized distance between image descriptors, most typically GIST. Autoencoder learning [20], on the other hand, learns a compact code which has good image reconstruction properties, but again is not specialized for category recognition. All the above descriptors can produce very compact codes, but few (excepting [24, 16]) have been shown to be effective at category-level recognition beyond simplified problems such as Caltech-20 [1] or Caltech-101 [12, 13]. In contrast, we consider Caltech-256 a baseline competence, and also test compact codes for the first time on an ImageNet [6] retrieval task.

The goal of this paper then is to learn a compact binary code (as short as 128 bits) which has good object-category recognition accuracy. In contrast to previous learning approaches, our training objective is a direct approximation to this goal; while in contrast to previous "designed" descriptors, we learn abstract categories (see figure 1) aimed at optimizing classification rather than an arbitrary predefined set of attributes or classemes, and thus achieve increased accuracy for a given code length.

2 Technical approach

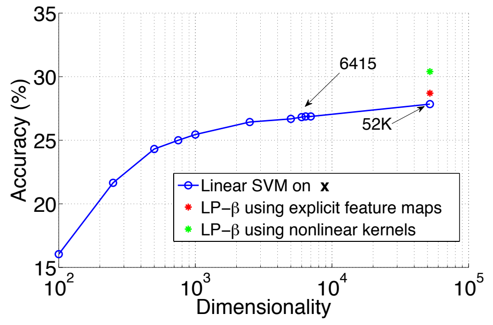

Figure 2: The accuracy versus compactness trade off. The benchmark is Caltech256, using 10 examples per class. The green dot shows the multiclass categorization accuracy achieved by an LP-$beta$ classifier using kernel distance; the red dot is the accuracy of an LP-$beta$ classifier that uses "lifted-up" features to approximate kernel distances; the blue line shows accuracy of a linear SVM trained on PCA projections of the lifted-up features, as a function of PCA dimension.

The basic classifier architecture used by state-of-the-art category recognizers, which we want to leverage as effectively as possible, uses image descriptors as follows. Given an image I, a bank of feature descriptors is computed (e.g. SIFT, PHOG, GIST), to yield a feature vector $\boldsymbol{f}_I \in \mathbb{R}^F$ (the feature vector used in our implementation has dimensionality $F = 17360$ and is described in the experimental section). State-of-the-art recognizers use kernel matching between these descriptors to define powerful classifiers, nonlinear in $\boldsymbol{f}_I$. For example, the LP-β classifier of Gehler and Nowozin [10], which has achieved the best results on several benchmarks to date, operates by combining the outputs of nonlinear SVMs trained on individual features. In our work we approximate these nonlinear classifiers by employing the lifting idea of Vedaldi and Zisserman [25]: for the family of homogeneous additive kernels K, there exists a finite-dimensional feature map $\hat{\psi} : \mathbb{R}^F \longrightarrow \mathbb{R}^{F(2r+1)}$ such that the nonlinear kernel distance $K(\boldsymbol{f}_I, \boldsymbol{f}_{I'}) \approx \left\langle \hat{\psi}(\boldsymbol{f}_I), \hat{\psi}(\boldsymbol{f}_{I'}) \right\rangle$ where r is a small positive integer (in our implementation set to 1). These explicit feature maps allow us to approximate a non-linear classifier, such as the LP-β kernel combiner, via an efficient linear projection. As described below, we intend to use these nonlinear classifier approximated via linear projections as the basis for learning our features. However, in our case $F(2r+1) = 17360 \times 3 = 52080$. This dimensionality is too large in practice for our learning. Thus, we perform a simple linear reduction $\boldsymbol{x}_I = [\mathbf{P}\boldsymbol{\phi}_I; 1]$ where projection matrix $\mathbf{P}$ is obtained through PCA, so $\boldsymbol{x}_I \in \mathbb{R}^n$ for $n \ll F(2r+1)$. Note that we append 1 to each $\boldsymbol{x}$ to avoid dealing with biases below. As this procedure is performed identically for every image we consider, we will drop the dependence on I and simply refer to the "image" $\boldsymbol{x}$.

A natural question to address is: how much accuracy do we lose due to the kernel approximation and the PCA projection? We answer this question in figure 2, where we compare the multi-class classification accuracies obtained on the Caltech256 data set by the following methods using our low-level descriptors $\boldsymbol{f}_I \in \mathbb{R}^{17360}$: an LP-$\beta$ combiner based on exact non-linear kernel-distance calculations; an LP-β combiner using explicit feature maps; a linear SVM trained on the PCA projections $\boldsymbol{x}_I$ as a function of the PCA subspace dimensionality. We see from this figure that the explicit maps degrade the accuracy only slightly, which is consistent with the results reported in [25]. However, the linear SVM produces slightly inferior accuracy even when applied to the full 52,080-dimensional feature vectors. The key-difference between the linear SVM and the LP-β classifier is that the former defines a classifier in the joint space of all 13 features, while the latter first trains a separate classifier for each feature, and then learns a linear combination of them. The results in our figure suggest that the two-step procedure of LP-β provides a form of beneficial regularization, a fact first noted in [10]. For our feature learning algorithm, we chose to use a PCA subspace of dimensionality $n = 6416$ since, as suggested by the plot, this setting gives a good tradeoff in terms of compact dimensionality and good recognition accuracy.

Torresani et al. [24] have shown that an effective image descriptor for categorization can be built by collecting in a vector the thresholded outputs of a large set of nonlinear classifiers evaluated on the image. This "classeme" descriptor can produce recognition accuracies within 10% of the state of the art for novel classes even with simple linear classification models. Using our formulation based on explicit feature maps, we can approximately express each classeme entry (which in [24] is implemented as an LP-β classifier) as the output of a linear classifier

$$h(\boldsymbol{x}; \boldsymbol{a}_c) = \mathbf{1}[\boldsymbol{a}_c^T \boldsymbol{x} > 0] \tag{1}$$

where $\mathbf{1}[.]$ is the 0-1 indicator function of its boolean argument. If following the approach of Torresani et al. [24], we would collect C training categories, and learn the parameters $\boldsymbol{a}_c$ for each from offline training data using some standard training objective such as hinge loss. We gather the parameters into a $n \times C$ matrix

$$\boldsymbol{A} = [\boldsymbol{a}_1 | \ldots | \boldsymbol{a}_C].$$

Then, for image $\boldsymbol{x}$, a new descriptor, the "classeme" descriptor $\boldsymbol{h}(\boldsymbol{x})$ is computed as the concatenation of the outputs of the training categories:

$$\boldsymbol{h}(\boldsymbol{x}; \boldsymbol{A}) = \begin{bmatrix} h(\boldsymbol{x}; \boldsymbol{a}_1) \\ \vdots \\ h(\boldsymbol{x}; \boldsymbol{a}_C) \end{bmatrix} \in \{0,1\}^C \tag{2}$$

The PICODE descriptor is of exactly this form, the only difference to [24] being our training procedure, and the fact that C is no longer restricted to being the same as the number of training classes.

To emphasize once more the contributions of this paper, let us review the shortcomings of existing descriptors, which we overcome in this paper

- Prior work used attributes learned disjointly from one another, which "just so-happen" to work well as features for classification, without theoretical justification for their use in subsequent classification. Given that we want to use attributes as features for linear classification, we propose to formalize as learning objective that linear combinations of such attributes must yield good accuracy.
- Unlike the attribute or classeme approach, our method decouples the number of training classes from the target dimensionality of the binary descriptor. We can optimize our features for any arbitrary desired length, thus avoiding a suboptimal feature selection stage.
- Finally, we directly optimize the learning parameters with respect to *binary* features while prior systems binarized the features in a final quantization stage.

We now introduce a framework that allows us to learn the $\boldsymbol{A}$ parameters directly on a multiclass classification objective.

2.1 Learning the basis classifiers

We assume that we are given a set of N training images, with each image coming from one of K training classes. We will continue to let C stand for the dimensionality (i.e. number of bits) of our code. Let $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N$ be the training set for learning the basis classifiers, where $\boldsymbol{x}_i$ is the i-th image example (represented by its n-dimensional PCA projection) and $\boldsymbol{y}_i \in \{-1,+1\}^K$ is a vector encoding the category label out of K possible classes: $y_{ik} = +1$ iff the i-th example belongs to class k.

We then define our c-th basis classifier to be a boolean function of the form (1), a thresholded *non-linear* projection of the original low-level features, parameterized by $\boldsymbol{a}_c \in \mathbf{R}^n$. We then optimize these basis classifiers so that linear combinations of these basis classifiers yield good categorization accuracy on $\mathcal{D}$. The learning objective introduces auxiliary variables $(\boldsymbol{w}_k, b_k)$ for each training class, which parameterize the linear classifier for that training class, operating on the PICODE representation of the training examples, and the objective for $\boldsymbol{A}$ simply minimizes over these auxiliaries:

$$E(\boldsymbol{A}) = \min_{\boldsymbol{w}_{1..K}, b_{1..K}} E(\boldsymbol{A}, \boldsymbol{w}_{1..K}, b_{1..K}) \tag{3}$$

Solving for $\boldsymbol{A}$ then amounts to simultaneous optimization over all variables of the following learning objective, which is a trade off between a small classification error and a large margin when using the output bits of the basis classifiers as features in a one-versus-all linear SVM:

$$E(\boldsymbol{A}, \boldsymbol{w}_{1..K}, b_{1..K}) = \sum_{k=1}^{K} \left\{ \frac{1}{2} \|\boldsymbol{w}_k\|^2 + \frac{\lambda}{N} \sum_{i=1}^{N} \ell \left[y_{i,k}(b_k + \boldsymbol{w}_k^\top \boldsymbol{h}(\boldsymbol{x}_i; \boldsymbol{A}) \right] \right\} \tag{4}$$

where $\ell[\cdot]$ is the traditional hinge loss function. Expanding, we get

$$E(\boldsymbol{A}, \boldsymbol{w}_{1..K}, b_{1..K}) = \sum_{k=1}^{K} \left\{ \frac{1}{2} \|\boldsymbol{w}_k\|^2 + \frac{\lambda}{N} \sum_{i=1}^{N} \ell \left[y_{i,k}(b_k + \sum_{c=1}^{C} w_{kc} \mathbf{1}[\boldsymbol{a}_c^T \boldsymbol{x}_i > 0]) \right] \right\}$$

Note that the linear SVM and the basis classifiers are learned jointly using the method described below.

2.2 Optimization

We propose to minimize this error function by block coordinate descent. We alternate between the two following steps:

1. Learn classifiers.
We fix $\boldsymbol{A}$ and optimize the objective with respect to $\boldsymbol{w}$ and $\boldsymbol{b}$ jointly. This optimization is convex and equivalent to traditional linear SVM learning.

2. Learn projectors.
Given the current values of $\boldsymbol{w}$ and $\boldsymbol{b}$, we minimize the objective with respect to $\boldsymbol{A}$ by updating one basis-classifier at a time. Let us consider the update of $\boldsymbol{a}_c$ with fixed parameters $\boldsymbol{w}_{1..K}, \boldsymbol{b}, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_{c-1}, \boldsymbol{a}_{c+1}, \ldots, \boldsymbol{a}_C$. It can be seen (Appendix A) that in this case the objective becomes:

$$E(\boldsymbol{a}_c) = \sum_{i=1}^{N} v_i \mathbf{1}[z_i \boldsymbol{a}_c^T \boldsymbol{x}_i > 0] + const \tag{5}$$

where $z_i \in \{-1, +1\}$ and $v_i \in \mathbb{R}^+$ are known values computed from the fixed parameters. Optimizing the objective in Eq. 5 is equivalent to learning a linear classifier minimizing the sum of weighted misclassifications, where v_i represents the cost of misclassifying example i. Unfortunately, this objective is not convex and it is difficult to optimize. Thus, we replace it with the following convex upper bound defined in terms of the hinge function ℓ:

$$\hat{E}(\boldsymbol{a}_c) = \sum_{i=1}^{N} v_i \ell(z_i \boldsymbol{a}_c^T \boldsymbol{x}_i) \tag{6}$$

This objective can be globally optimized using an LP solver or software for SVM training. We had success with liblinear [8], dealing with the large problem sizes encountered in both optimization steps.

We have also experimented with several other optimization methods, including stochastic gradient descent applied to a modified version of our objective where we replaced the binarization function $h(\boldsymbol{x}; \boldsymbol{a}_c) = \mathbf{1}[\boldsymbol{a}_c^T \boldsymbol{x} > 0]$ with the sigmoid function $\sigma(\boldsymbol{x}; \boldsymbol{a}_c) = 1/(1 + \exp(-\frac{2}{T} \boldsymbol{a}_c^T \boldsymbol{x}))$ to relax the problem. After learning, at test time we replaced back $\sigma(\boldsymbol{x}; \boldsymbol{a}_c)$ with $h(\boldsymbol{x}; \boldsymbol{a}_c)$ to obtain binary descriptors. However, we found that these binary codes performed much worse than those directly learned via the coordinate descent procedure described above.

3 Experiments

We now describe experimental evaluations carried out over several data sets. In order to allow a fair comparison, we reimplemented the "classeme descriptor" based on the same set of low-level features and settings described in [24] but using the explicit feature map framework to replace the expensive nonlinear kernel distance computations. The low-level features are: color GIST [18], spatial pyramid of histograms of oriented gradients (PHOG) [3], spatial pyramid of self-similarity descriptors [22], and a histogram of SIFT features [17] quantized using a dictionary of 5000 visual words. Each spatial pyramid level of each descriptor was treated as a separate feature, thus producing a total of 13 low-level features. Each of these features was lifted up to a higher-dimensional space using the explicit feature maps of Vedaldi and Zisserman [25]. We chose the mapping approximating the histogram intersection kernels for $n = 1$, which effectively mapped each low-level feature descriptor to a space 3 times larger than its original one. This produced vectors $\boldsymbol{\phi}$ having dimensionality $3 \times F = 52,080$. To learn our basis classifiers, we used 6416-dimensional PCA projections of these high-dimensional vectors.

We learned PiCoDes with the approach described here and compared them to binary classeme vectors. In [24], classemes have been shown to clearly outperform classifiers trained on individual

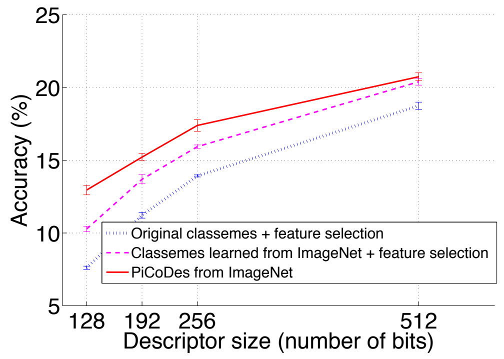

Figure 3: Multiclass categorization accuracy on Caltech256 using different binary codes, as a function of the number of bits. Although PICODES were learned from only 1/5-th of the data used to train classemes, PICODES provide consistently better categorization accuracy.

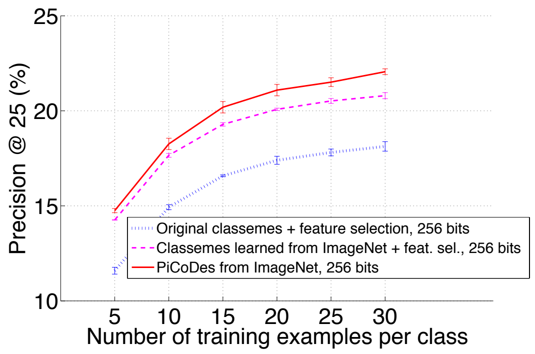

Figure 4: Precision of object-class search using 256-bit codes on Caltech256: for a varying number of training examples per class, we report the percentage of true positives in top 25 retrieved from a dataset containing 6375 distractors and 25 relevant results.

low-level features, such as GIST. The categorization accuracy of binary codes that approximate low-level features (e.g., [23]) would be even lower and thus we do not consider them in our comparison. We trained both PICODES and classeme vectors using a training set of $K = 2625$ classes randomly sampled from the ImageNet dataset [6]. Each class in ImageNet is associated to a "synset", which is a set of words describing the category. Since we wanted to evaluate the learned descriptors on the Caltech256 benchmark, we selected the 2625 ImageNet training classes such that the synsets of these classes do not contain any of the Caltech256 class labels, so as to avoid "pre-learning" the test classes during the feature-training stage. To train the classeme classifiers we used 150 images for each training category, for a total of $N = 2625 \times 150 = 393,750$ examples, thus reproducing the learning setup described in [24]. We learned the 2625 classeme classifiers in parallel using a powerful cluster of machines. However, our algorithm solves jointly for all C features and consequently the computation cannot be easily parallelized. Thus, for our approach we reduced the number of examples per class to be only 30, for a total of $N = 2625 \times 30 = 78,750$ images.

Multiclass recognition using PICODES. The first experiment is to examine the claim that PICODES expand the compactness/accuracy envelope. This is illustrated in figure 5, where the new codes are significantly more compact than was previously possible, with a reasonable reduction in accuracy (e.g. an accuracy of 28% for classemes at 180 bytes per image compared to 23% for PICODES at 64 bytes per image, each with 15 training examples).

We first report results showing multiclass classification accuracy achieved with binary codes on the Caltech256 data set. Since PICODES are optimized for categorization using linear models, we adopt simple linear "one-versus-all" SVMs as classifiers. For each Caltech256 category, the classifier was trained using 10 positive examples and a total of 2550 negative examples obtained by sampling 10 images from each of the other classes. We computed accuracies averaged over five random splits of training and test data, using 25 test examples per class in each split. As usual accuracy is computed as the average over the mean recognition rates per class. Figure 3 shows the results obtained with binary descriptors of varying dimensionality. While our approach can accommodate easily the case were the number of feature dimensions (C) is different from the number of feature-training categories (K), the classeme learning method can only produce descriptors of size K. Thus, the descriptor size is typically reduced through a subsequent feature selection stage [24, 16]. In this figure we show accuracy obtained with classeme features selected by multi-class recursive feature elimination with SVM [4], which at each iteration retrains the SVMs for all classes on the active features and then removes the m active features that are least used by the classifiers until reaching the desired compactness. We also report accuracy obtained with the original classeme vectors of [24], which were learned on weakly-supervised images retrieved with text-base image search engines. Although our approach uses only 1/5-th of the training examples used to learn classemes, PICODES provide a significant improvement in accuracy over the classeme descriptors, particularly when the

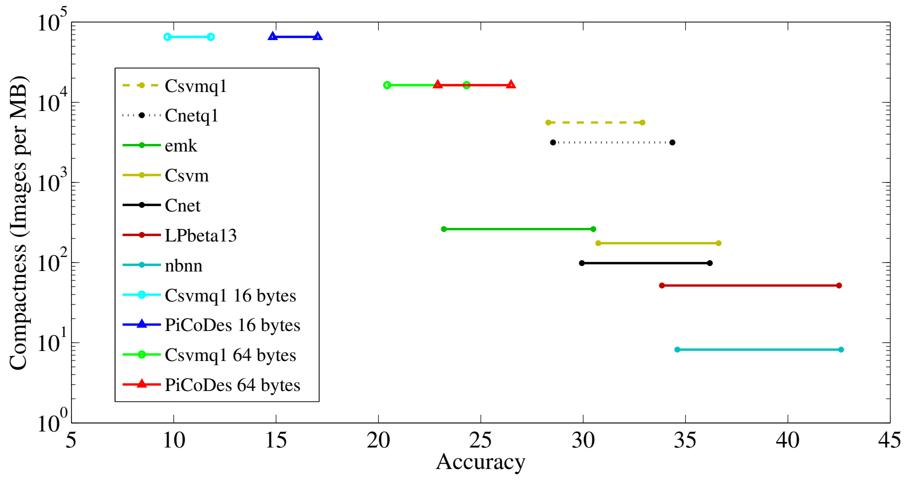

Figure 5: **Compactness/accuracy envelope.** Various existing methods are plotted in terms of the compactness of representation versus multiclass accuracy on Caltech-256. On both axes, higher is better. (Note logarithmic y-axis). The lines link performance at 15 and 30 training examples. See [24] for details of the non-PICODE algorithms. The PICODE descriptor offers significant increases in compactness, with reasonable accuracy reductions.

target dimensionality is very small. Note that our 512-bit codes approach the accuracy of the original full-size classeme vectors (20.7% and 24.5% error, respectively) while being 5 times more compact.

Retrieval of object classes on Caltech256. Next we present results corresponding to our motivating application of object-class search, using 256-bit PICODES. For each Caltech256 class, we trained a one-versus-all linear SVM using p positive examples and $p \times 255$ negative examples, for varying values of p. We then used the learned classifier to find images of that category in a database containing 6400 Caltech256 test images, with 25 images per class. The retrieval accuracy is measured as precision at 25, which is the proportion of true positives (i.e., images of the query class) ranked in the top 25. Again, we see that our features yield consistently better ranking precision compared to classeme vectors learned on the same ImageNet training set, and produce an improvement of about 4% over the original classeme features.

Object class search in a large image collection. Finally, we present experiments on the 150K image data set of the Large Scale Visual Recognition Challenge 2010 (ILSVRC2010) [2], which includes images of 1000 different categories. In order to avoid a biased evaluation, for this experiment we removed from the $K = 2625$ ImageNet training classes used to learn PICODES, all categories that also appear in the ILSVRC2010 data set. We found an overlap of 357 classes between these sets, which we eliminated, and then we re-learned PICODES on the remaining $K = 2625 - 357 = 2268$ classes. Again, we evaluate our binary codes on the task of object-class retrieval. For each of the 1000 classes, we train a linear SVM using all examples of that class available in the ILSVRC2010 training set (this number varies from a minimum of 619 to a maximum of 3047, depending on the class) and 4995 negative examples obtained by sampling five images from each of the other classes. We test the classifiers on the ILSVRC2010 test set, which includes 150 images for each of the 1000 classes. Figure 6 shows a comparison between PICODES

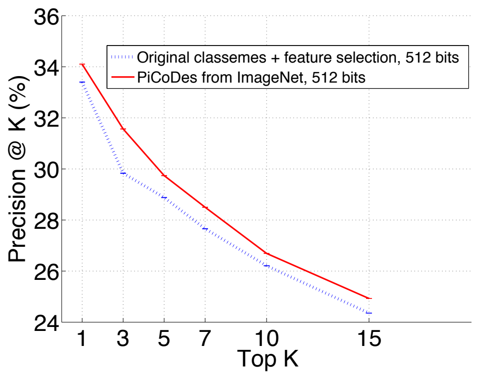

Figure 6: Finding pictures of an object class in the ILSVRC2010 dataset, which includes 150K images for 1000 different classes. PICODES enable accurate class retrieval from this large collection in less than a second.

and classemes in terms of precision at k for varying k. Despite the very large number of distractors (149,850 for each query), search with our features yields precisions exceeding 30%. Furthermore, the tiny size of our descriptor allows the entire data set to be easily kept in memory for efficient retrieval (the whole database size using our representation is only 10MB): the average search time for a query class, including the learning time, is about 1 second on an Intel Xeon X5680 @ 3.33GHz.

4 Conclusion

We have described a new type of compact code, which is learned by directly minimizing a multi-class classification objective on an object category recognition problem. This allows category-level recognition tasks to be performed using extremely compact codes. Although there is much existing work on learning compact codes, we know of no other code which offers this performance on a category recognition task. Of course, the absence of free lunch is certainly evident in our results. The key illustration is in figure 5, where the ideal result would be new operating points in the top right of the figure: a compact code which also offers high performance. It may be that "attributes" based classifiers will one day occupy that space, but to date, their training and evaluation has not been automated enough to make that claim. This paper's contribution is to populate the top of the graph, expanding the compactness/accuracy envelope.

Our experiments have focussed on whole-image "Caltech-like" category recognition, while it is clear that subimage recognition is also an important application. However, we argue that for many image search tasks, whole-image performance is relevant, and for a very compact code, one could possibly encode several windows (dozens, say) in each image, while retaining a relatively compact representation.

A Derivation eq. 5

We present below the derivation of eq. 5. First, we rewrite our objective function, i.e., eq. 4, in expanded form:

$$E(\boldsymbol{A}, \boldsymbol{w}_{1..K}, b_{1..K}) = \sum_{k=1}^{K} \left\{ \frac{1}{2}\|\boldsymbol{w}_k\|^2 + \frac{\lambda}{N}\sum_{i=1}^{N} \ell\left[y_{ik}(b_k + \sum_{c=1}^{C} w_{kc}\mathbf{1}[\boldsymbol{a}_c^T\boldsymbol{x}_i > 0]) \right] \right\}.$$

Fixing the parameters $\boldsymbol{w}_{1..K}, \boldsymbol{b}, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_{c-1}, \boldsymbol{a}_{c+1}, \ldots, \boldsymbol{a}_C$ and minimizing the function above with respect to $\boldsymbol{a}_c$, is equivalent to minimizing the following objective:

$$E'(\boldsymbol{a}_c) = \sum_{k=1}^{K}\sum_{i=1}^{N} \ell\left[y_{ik}w_{kc}\mathbf{1}[\boldsymbol{a}_c^T\boldsymbol{x}_i > 0] + y_{ik}b_k + \sum_{c'\neq c} y_{ik}w_{kc'}\mathbf{1}[\boldsymbol{a}_{c'}^T\boldsymbol{x}_i > 0] \right].$$

Let us define $\alpha_{ikc} \equiv y_{ik}w_{kc}$, and $\beta_{ikc} \equiv (y_{ik}b_k + \sum_{c'\neq c} y_{ik}w_{kc'}\mathbf{1}[\boldsymbol{a}_{c'}^T\boldsymbol{x}_i > 0])$. Then, we can rewrite the objective as follows:

$$\begin{aligned} E'(\boldsymbol{a}_c) &= \sum_{k=1}^{K}\sum_{i=1}^{N} \ell\left[\alpha_{ikc}\mathbf{1}[\boldsymbol{a}_c^T\boldsymbol{x}_i > 0] + \beta_{ikc} \right] \\ &= \sum_{i=1}^{N} \left\{ \mathbf{1}[\boldsymbol{a}_c^T\boldsymbol{x}_i > 0] \sum_{k=1}^{K} \ell(\alpha_{ikc} + \beta_{ikc}) + (1 - \mathbf{1}[\boldsymbol{a}_c^T\boldsymbol{x}_i > 0]) \sum_{k=1}^{K} \ell(\beta_{ikc}) \right\} \\ &= \sum_{i=1}^{N} \left\{ \mathbf{1}[\boldsymbol{a}_c^T\boldsymbol{x}_i > 0] \sum_{k=1}^{K} \ell(\alpha_{ikc} + \beta_{ikc}) - \ell(\beta_{ikc}) \right\} + const\,. \end{aligned}$$

Finally, it can be seen that optimizing this objective is equivalent to minimizing

$$E(\boldsymbol{a}_c) = \sum_{i=1}^{N} v_i \mathbf{1}[z_i \boldsymbol{a}_c^T\boldsymbol{x}_i > 0]$$

where $v_i = \left| \sum_{k=1}^{K} \ell(\alpha_{ikc} + \beta_{ikc}) - \ell(\beta_{ikc}) \right|$ and $z_i = \text{sign}\left(\sum_{k=1}^{K} \ell(\alpha_{ikc} + \beta_{ikc}) - \ell(\beta_{ikc}) \right)$. This yields eq. 5.

References

[1] B. Babenko, S. Branson, and S. Belongie. Similarity metrics for categorization: From monolithic to category specific. In *Intl. Conf. Computer Vision*, pages 293 –300, 2009.

[2] A. Berg, J. Deng, and L. Fei-Fei. Large scale visual recognition challenge, 2010. http://www.image-net.org/challenges/LSVRC/2010/.

[3] A. Bosch, A. Zisserman, and X. Muñoz. Representing shape with a spatial pyramid kernel. In *Conf. Image and Video Retrieval (CIVR)*, pages 401–408, 2007.

[4] O. Chapelle and S. S. Keerthi. Multi-class feature selection with support vector machines. *Proc. of the Am. Stat. Assoc.*, 2008.

[5] O. Chum, J. Philbin, and A. Zisserman. Near duplicate image detection: min-hash and tf-idf weighting. In *British Machine Vision Conf.*, 2008.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.

[7] M. Douze, A. Ramisa, and C. Schmid. Combining attributes and fisher vectors for efficient image retrieval. In *Proc. Comp. Vision Pattern Recogn. (CVPR)*, 2011.

[8] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[9] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing objects by their attributes. In *CVPR*, 2009.

[10] P. Gehler and S. Nowozin. On feature combination for multiclass object classification. In *ICCV*, 2009.

[11] A. G. Hauptmann, R. Yan, W.-H. Lin, M. G. Christel, and H. D. Wactlar. 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.

[12] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *Proc. Comp. Vision Pattern Recogn. (CVPR)*, 2010.

[13] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. In *Intl. Conf. Computer Vision*, 2010.

[14] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar. Attribute and Simile Classifiers for Face Verification. In *Intl. Conf. Computer Vision*, 2009.

[15] C. H. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*, 2009.

[16] L. Li, H. Su, E. Xing, and L. Fei-Fei. Object Bank: A high-level image representation for scene classification & semantic feature sparsification. In *NIPS*. 2010.

[17] D. Lowe. Distinctive image features from scale-invariant keypoints. *Intl. Jrnl. of Computer Vision*, 60(2):91–110, 2004.

[18] A. Oliva and A. Torralba. Building the gist of a scene: The role of global image features in recognition. *Visual Perception, Progress in Brain Research*, 155, 2006.

[19] M. Raginsky and S. Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.

[20] M. Ranzato, Y. Boureau, and Y. LeCun. Sparse feature learning for deep belief networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.

[21] R. Salakhutdinov and G. Hinton. Semantic hashing. *Int. J. Approx. Reasoning*, 50:969–978, 2009.

[22] E. Shechtman and M. Irani. Matching local self-similarities across images and videos. In *Proc. Comp. Vision Pattern Recogn. (CVPR)*, 2007.

[23] A. Torralba, R. Fergus, and Y. Weiss. Small codes and large image databases for recognition. In *Proc. Comp. Vision Pattern Recogn. (CVPR)*, 2008.

[24] L. Torresani, M. Szummer, and A. Fitzgibbon. Efficient object category recognition using classemes. In *ECCV*, 2010.

[25] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. In *CVPR*, 2010.

[26] J. Vogel and B. Schiele. Semantic modeling of natural scenes for content-based image retrieval. *Intl. Jrnl. of Computer Vision*, 72(2):133–157, 2007.

[27] G. Wang, D. Hoiem, and D. Forsyth. Learning image similarity from flickr using stochastic intersection kernel machines. In *Intl. Conf. Computer Vision*, 2009.

[28] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*. 2009.