Classemes and Other Classifier-based Features for Efficient Object Categorization

Alessandro Bergamo, and Lorenzo Torresani, *Member, IEEE*

Abstract—This paper describes compact image descriptors enabling accurate object categorization with linear classification models, which offer the advantage of being efficient to both train and test. The shared property of our descriptors is the use of classifiers to produce the features of each image. Intuitively, these classifiers evaluate the presence of a set of *basis classes* inside the image. We first propose to train the basis classifiers as recognizers of a hand-selected set of object classes. We then demonstrate that better accuracy can be achieved by learning the basis classes as "abstract categories" collectively optimized as features for linear classification. Finally, we describe several strategies to aggregate the outputs of basis classifiers evaluated on multiple subwindows of the image in order to handle cases when the photo contains multiple objects and large amounts of clutter. We test our descriptors on challenging benchmarks of object categorization and detection, using a simple linear SVM as classifier. Our results are on par with those achieved by the best systems in these fields but are produced at orders of magnitude lower computational costs and using an image representation that is general and not specifically tuned for a predefined set of test classes.

Index Terms—Object categorization, image features, attributes.

1 Introduction

In this work we consider the problem of object class recognition in large image databases. Over the last few years this topic has received a growing amount of attention in the vision community [1], [2]. We argue, however, that nearly all proposed systems have focused on a scenario involving two restrictive assumptions: the first, is that the recognition problem involves a *fixed* set of categories, known before the creation of the database; the second, is that there are no constraints on the learning and testing time of the object classifiers, besides the requirement that training and testing must be feasible. Such assumptions are reflected in the characteristics of the most popular object categorization benchmarks [3], [4], which measure the performance of recognition systems solely in terms of classification accuracy over a *predefined* set of classes, and without consideration of the computational costs.

We believe that these two assumptions do not meet the requirements of modern applications of large-scale object categorization. For example, test-recognition efficiency is a fundamental requirement to be able to scale object classification to Web photo repositories, such as Flickr, which are growing at rates of several millions new photos per day. Furthermore, while a fixed set of object classifiers can be used to annotate pictures with a set of predefined tags, the interactive nature of searching and browsing large image collections calls for the ability to allow users to define their own personal query categories to be recognized and retrieved from the database, ideally in real-time. Depending on the application, the user can 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. 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.

In this paper we consider the problem of designing a system that can address these requirements: our goal is to develop an approach that enables accurate *real-time recognition* of arbitrary categories in gigantic image collections, where the *classes are not defined in advance*, i.e, they are not known at the time of the creation of the database. We propose to achieve this goal by means of image descriptors designed to enable good recognition accuracy with simple linear classifiers, which can be trained efficiently and – perhaps even more crucially – can be tested in just a few seconds even on databases containing millions of images. Rather than optimizing classification accuracy for a fixed set of classes, our aim is to learn a general image representation which can be used to describe and recognize arbitrary categories, even novel classes not present in the training set used to learn the descriptor.

We propose to use as entries of our image descriptor the outputs of a predefined set of *nonlinear* classifiers evaluated on low-level features computed from the photo. This implies that a simple linear classification model applied to this descriptor effec-

A. Bergamo and L. Torresani are with the Department of Computer Science, Dartmouth College, Hanover, NH 03755, U.S.A.
E-mail: {aleb, lorenzo}@cs.dartmouth.edu

tively implements a nonlinear function of the original low-level features. As demonstrated in recent literature on object categorization [5], these nonlinearities are critical to achieve good categorization accuracy with low-level features. However, the advantage of our approach is that our classification model, albeit nonlinear in the low-level features, remains *linear* in our descriptor and thus it enables efficient training and testing. In other words, the nonlinear classifiers implementing our features are used as a classification basis to recognize new categories via linear models. Based on this intuition, we refer to our features as *basis classes*. The final classifier for a novel class is obtained by linearly combining the outputs of the nonlinear classifiers, which we can pre-compute and store for every image in the database, thus enabling efficient novel-class recognition even in large datasets.

In this work we investigate and propose several strategies to define and train the basis classifiers. In section §3.2 we first propose to use as basis classes a hand-selected set of object classes from the real-world: in this case each basis classifier is simply trained as a traditional object classification model optimized to recognize a particular object category. This idea is evocative of the use of attributes [6], [7], [8] which are fully-supervised classifiers trained to recognize certain properties in the image such as "has beak", "near water". While attributes have been used as features for recognition in specialized domains (e.g., animal recognition [8] or face identification [6]), we demonstrate that by choosing a large and varied set of object categories as basis classes, it is possible to employ the resulting descriptor as an effective universal feature representation for general object categorization. Furthermore, we show that this feature vector can be binarized with little loss of recognition accuracy to produce a compact binary code that allows even gigantic image collections to be kept in memory for more efficient testing.

However, in this work we demonstrate that a better classifier-based representation can be obtained by learning the basis classes as general "abstract categories" optimized to yield useful features for linear models rather than hand-defining them as real object classes. We propose two distinct strategies to learn automatically the basis classes: the first (§3.3) optimizes the basis classifiers for linear classification, i.e., it trains them to produce good recognition accuracy when used as features with linear models; the second strategy (§3.4) constrains each basis class to be a super-category obtained by grouping a set of object classes such that, collectively, they are easy to distinguish from other sets of categories. When compared to existing work on learning compact image codes, we show that our feature vectors provide better accuracy for the same descriptor size.

While multiclass recognition of a fixed set of categories is *not* our main motivating application, we show that our approaches achieve excellent performance even on this task. On the Caltech256 benchmark, a simple linear SVM trained on our representation outperforms the state-of-the-art LP-β classifier [5] trained on the same low-level features used to learn our descriptor. On the 2010 ImageNet Challenge (ILSVRC2010) database, linear classification with our features achieves recognition accuracy only 10.3% lower than the winner of the competition [9], whose computational cost for training and testing is several orders of magnitude higher compared to our approach.

Finally, we present the first comprehensive evaluation of classifier-based descriptors on object detection and scene recognition datasets. In order to render the descriptor capable of handling the presence of multiple objects and clutter typical of these datasets, we propose to apply our basis classifiers on many subwindows of the photo. We describe and test several strategies to aggregate the features produced from multiple subwindows into a single compact descriptor that can be used by linear classification models. We show that the resulting descriptors yield a significant boost in accuracy compared to feature vectors built from basis classifiers evaluated on the whole image and are on par with specialized object detectors tuned on the test classes. However, the training and testing of linear classifiers using our descriptors have considerably lower computational cost.

2 Related work

The problem of object class recognition in large datasets has been the subject of much recent work. While nonlinear classifiers are recognized as the state-of-the-art in terms of categorization accuracy [5], [10], they are difficult to scale to large training sets. Thus, much more efficient linear models are typically adopted in recognition settings involving a large number of object classes, with many image examples per class [1]. As a result, much work in the last few years has focused on methods to retain high recognition accuracy even with linear classifiers. We can loosely divide these methods in three categories.

The first category comprises techniques to approximate nonlinear kernel distances via explicit feature maps [11], [12]: for many popular kernels in computer vision, these methods provide analytical mappings to higher-dimensional feature spaces where inner products approximate the kernel distance. This permits to achieve results comparable to those of the best nonlinear classifiers with simple linear models. However, these approaches are difficult to use when the training and test sets are very large, due to the high storage costs caused by the dimensionality of the data in the "lifted-up" space.

A second line of work involves the use of high-dimensional feature vectors to produce a higher degree of linear separability [2]: this idea is similar to

the one behind the use of explicit feature maps, with the difference that these high-dimensional signatures are not produced with the intent of approximating kernel distances between lower-dimensional features but rather to yield higher accuracy with linear models. Since large datasets represented with these high-dimensional descriptors cannot be kept in memory, the feature vectors are often stored in compressed form and they are decompressed on the fly "one at a time" during training and testing [2], [13].

Finally, the third strand of related work involves the use of image descriptors encoding categorical information as features: the image is represented in terms of its relation to a set of basis object classes [14], [15], [16] or as the response map to a set of detectors [17]. Even linear models applied to these high-level representations have been shown to produce good categorization accuracy. These descriptors can be viewed as generalizing attributes [6], [7], [8], which are semantic characteristics selected by humans as associated to the classes to recognize.

Our three approaches presented in this paper are closely related to this third line of work, as they all represent images in terms of the outputs of classifiers learned for a set of basis classes. Our first approach – classemes – (§3.2) was originally presented in [15] and extends attributes to work on broad object-class recognition by using a large set of real object classes as features. In section §3.3 instead we describe PiCoDes (first introduced in [18]), which are binary descriptors learned to explicitly maximize linear classification accuracy on a large set of training object classes. The third method – meta-classes – which we originally presented in [19] is described in section §3.4. This method learns the basis classes as a set of abstract categories capturing useful properties for object class recognition; the learning of these abstract categories is much more efficient and scalable than in the case of PiCoDes and it enables training of descriptors of much larger size.

In this article we also consider how to extend classifier-based image descriptors to yield good recognition accuracy even when multiple objects and clutter is present in the photo. Our approach is inspired by Li et al. [17] who have proposed to use the localized outputs of object detectors as image features, to encode spatial information into a global image descriptor. However, this representation is very high-dimensional (the "ObjectBank" descriptor includes over 40,000 real-valued entries) and as such is not suited for recognition in large databases. Here we present several strategies to aggregate the outputs of basis classifiers evaluated on multiple subwindows in a single low-dimensional descriptor. We show that simple linear classifiers trained on our aggregate vector produce results approaching the state-of-the-art on challenging object-detection and scene recognition benchmarks.

3 Technical Approach

3.1 General framework

In this section we introduce our general framework of classifier-based image descriptors. At a high-level, our approach involves representing an image $\boldsymbol{x}$ as a C-dimensional vector $\boldsymbol{h}(\boldsymbol{x})$, where the c-th entry is the output of a classifier h_c evaluated on $\boldsymbol{x}$:

$$\boldsymbol{h}(\boldsymbol{x}) = \begin{bmatrix} h_1(\boldsymbol{x}) \\ \vdots \\ h_C(\boldsymbol{x}) \end{bmatrix} \quad (1)$$

The classifiers $h_{1\ldots C}$ (the basis classifiers) are learned during an offline stage from a large labeled dataset of images $\mathcal{D}^S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, where $\boldsymbol{x}_i$ denotes the i-th image in the database and $y_i \in \{1, \ldots, K\}$ indicates the class label of the object present in the photo. The database $\mathcal{D}^S$ represents our general visual knowledge-base; it should be very large and ideally include all possible visual concepts of the world. Our approaches analyze $\mathcal{D}^S$ to automatically extract the C visual classifiers $h_{1\ldots C}$ to be subsequently used as universal image features for class recognition. We call these C visual features basis classes. Note that the basis classes may be abstract categories, i.e., visual categories that do not necessarily exist in the real-world but that are useful to represent and discriminate the K classes in $\mathcal{D}^S$.

Before describing how we define and learn the basis classifiers, we want to discuss the motivation behind this representation. Intuitively, our descriptor represents an image in terms of its visual closeness to the set of C basis classes. We propose to extract this descriptor for every image $\boldsymbol{x}$ in the database where we want to perform the final recognition. As shown in our experiments, these classifier-based representation provides us with two fundamental advantages: 1) Compactness. Only C entries are are needed to describe each image. By limiting C to relatively small values (say, a few thousands), we can store even large image collections in memory (rather than on the hard-drive) for more efficient recognition. 2) High accuracy with linear models. This representation enables good classification accuracy even with simple linear models of the form $\boldsymbol{\theta}^\top \boldsymbol{h}(\boldsymbol{x})$. Intuitively this happens because a linear combination of these C features implements a powerful weighted sum of C classifiers, which reuses the basis classes as building blocks to recognize novel categories.

Let us now consider the problem of how to define the basis classifiers. We propose to train our basis classifiers on a low-level representation of the image: given an image $\boldsymbol{x}$ we first extract a set of M low-level features $\{\boldsymbol{f}_m(\boldsymbol{x})\}_{m=1}^M$, capturing different visual cues, such as the color distribution, or the spatial layout in the image (our low-level features include common descriptors such as SIFT [20] and GIST [21]; see sec.4.2 for further details). In order to obtain a powerful

image representation, we want our basis classifiers $h_{1...C}$ to be *nonlinear* functions of features $\{\boldsymbol{f}_m(\boldsymbol{x})\}$. This is motivated by the observation that several recent papers have shown that such nonlinearities are crucially necessary to achieve good classification accuracy (see, e.g., [5], [15]). However, at the same time we want to define a representation that is efficient to compute. This implies that the basis classifiers $h_{1...C}$ must be fast to evaluate. In order to achieve this twofold purpose, we lift each feature vector $\boldsymbol{f}_m(\boldsymbol{x}) \in \mathbb{R}^{d_m}$ to a higher-dimensional feature space via the explicit map proposed in [12] so that inner products in this space approximate well a predefined nonlinear kernel distance $K_m()$ over the original features $\boldsymbol{f}_m(\boldsymbol{x})$. Formally, the explicit map Ψ_m for feature m is defined as a function $\Psi_m : \mathbb{R}^{d_m} \longrightarrow \mathbb{R}^{d_m(2r+1)}$ where $r \in \mathbb{Z}^+$ such that $\langle \Psi_m(\boldsymbol{f}_m(\boldsymbol{x})), \Psi_m(\boldsymbol{f}_m(\boldsymbol{x}'))\rangle \approx K_m(\boldsymbol{f}_m(\boldsymbol{x}), \boldsymbol{f}_m(\boldsymbol{x}'))$. For the family of additive kernels the map can be analytically computed. The parameter r is a positive integer parameter controlling the target dimensionality. It has been shown in [12] that good approximations can be obtained even when setting r to very small values (in this work we use $r = 1$; for more details see Sec. 4.2). This trick allows us to approximate a computationally-expensive kernel-based classifier with a linear model that can be trained and evaluated very efficiently. The concatenation of the M lifted-up features produces a vector $\Psi(\boldsymbol{x})$ of dimensionality $D = \sum_{m=1}^{M} d_m(2r+1)$:

$$\Psi(\boldsymbol{x}) = [\Psi_1(\boldsymbol{f}_1(\boldsymbol{x}))^\top, \ldots, \Psi_M(\boldsymbol{f}_M(\boldsymbol{x}))^\top]^\top \quad (2)$$

Using this formulation based on explicit feature maps, we implement each basis classifier h_c as a linear combination of approximate kernels parameterized by a single projection vector $\boldsymbol{a}_c$:

$$h_c(\boldsymbol{x}) = \tau_c(\boldsymbol{a}_c^\top [\Psi(\boldsymbol{f}(\boldsymbol{x})); 1]) \quad (3)$$

where τ_c is a function to either quantize or scale the classifier output. The constant value 1 is appended to the vector $\Psi(\boldsymbol{x})$ to implement a bias coefficient without explicitly keeping track of it. We can then stack together the parameters of all basis classifiers in a single matrix $\boldsymbol{A} = [\boldsymbol{a}_1|\ldots|\boldsymbol{a}_C]$. For notational convenience, we set the following equivalences: $h_c(\boldsymbol{x}) \equiv h(\boldsymbol{x}; \boldsymbol{a}_c) = \tau_c(\boldsymbol{a}_c^\top [\Psi(\boldsymbol{f}(\boldsymbol{x})); 1])$ and $\boldsymbol{h}(\boldsymbol{x}) \equiv \boldsymbol{h}(\boldsymbol{x}; \boldsymbol{A})$.

We can now even more clearly recognize the two above-mentioned advantages enabled by our descriptor, i.e, compactness and high accuracy with linear models. Specifically, this formulation implies that a simple linear classification model $\boldsymbol{\theta}^\top \boldsymbol{h}(\boldsymbol{x})$ trained for a new class effectively implements an approximate linear combination of multiple low-level kernel distances, using our basis classifiers as individual components:

$$\boldsymbol{\theta}^\top \boldsymbol{h}(\boldsymbol{x}) = \sum_{c=1}^{C} \theta_c \tau_c(\boldsymbol{a}_c^\top [\Psi(\boldsymbol{f}(\boldsymbol{x})); 1]) \quad (4)$$

The great advantage is that the model $\boldsymbol{\theta}$, albeit nonlinear in the low-level features, remains linear in our descriptor and thus can be efficiently trained and tested. Moreover, note that while the extraction of the low-level features $\{\boldsymbol{f}_m(\boldsymbol{x})\}_{m=1}^{M}$ is needed to calculate the descriptor $\boldsymbol{h}(\boldsymbol{x})$, the storage of these low-level features is not necessary for the subsequent recognition. This implies that only the compact feature vectors $\boldsymbol{h}(\boldsymbol{x})$ need to be stored in the database.

Our approach can also be viewed as a dimensional reduction method: the function $\boldsymbol{h}$ indeed maps the high-dimensional descriptor represented by Eq. 2, to a space of much lower dimensionality, i.e. $C \ll D$. In the experimental section we will show that such representation is able to retain or even enrich the original representation, allowing efficient linear classifiers to achieve state-of-the-art classification accuracy on several challenging benchmarks.

In the next sections we propose different methods to discover the basis classes and learn the basis classifiers from a given database $\mathcal{D}^S$.

3.2 Classemes

CLASSEMES are the classifier-based features that we first introduced in [15]. In this approach, the basis classes are defined directly as the real-world object categories of the labeled training set $\mathcal{D}^S$ (thus $C = K$). Each basis classifier h_c is a 1-vs-the-rest binary classifier trained to recognize the c-th category. Specifically, we train h_c using a training set $\mathcal{D}_c^S$ that contains the images of the c-th category as positive examples, and a random subset of images sampled from the other $K - 1$ classes as negative examples. In this work we implement the CLASSEMES $h_c(\boldsymbol{x})$ using the LP-β classification model [5], which has been shown to yield state-of-the-art results on several categorization benchmarks. The LP-β classifier is defined as a linear combination of M nonlinear SVMs, each trained on a different low-level feature vector $\boldsymbol{f}_m(\boldsymbol{x})$. While [5], [15] employ exact kernels to render the classifier nonlinear, here we use approximate kernel distances by adopting the explicit feature map described in the Sec. 3.1. As already noted, this modification speeds up the training and the evaluation of the basis classifiers by orders of magnitudes and thus enables a very efficient extraction of CLASSEMES. Formally, each basis classifier $h_c(\boldsymbol{x})$ is defined as:

$$h_c(\boldsymbol{x}) = \tau_c\left(\sum_{m=1}^{M} \beta_{m,c} \left[\boldsymbol{w}_{m,c}^T \Psi_m(\boldsymbol{f}_m(\boldsymbol{x})) + b_{m,c}\right]\right) \quad (5)$$

Note that by re-arranging the terms in Eq. 5, we can expresses h_c in the same form as Eq. 3. Following the customary training scheme of LP-β, we first learn the parameters $\{\boldsymbol{w}_{m,c}, b_{m,c}\}$ for each feature m independently by training the hypothesis $h_{m,c}(\boldsymbol{x}) = \left[\boldsymbol{w}_{m,c}^T \Psi_m(\boldsymbol{f}_m(\boldsymbol{x})) + b_{m,c}\right]$ using the traditional SVM

large-margin objective on training set $\mathcal{D}_c^S$. In a second step, we optimize over parameter vector $\boldsymbol{\beta}_c = [\beta_{1,c}, \ldots, \beta_{m,c}]^T$ subject to the constraint $\sum_m \beta_{m,c} = 1$. This last optimization can be shown to reduce to a simple linear program (see [5] for further details). An advantage of this training procedure is that it is embarassingly parallel, as each hypothesis h_c can be learned independently from the others. This fact makes this method very scalable to the size of the training database $\mathcal{D}^S$.

The functions τ_c in Eq. 5 are quantizers applied after the learning to reduce the dimensionality of the descriptor. In our original work [15] we evaluated different levels of quantization, realizing different trade-offs between the descriptor compactness and the final accuracy of the visual recognition system. In this work, we consider the following two options:

- CLASSEMES We use the raw outputs of the LP-β classifiers as features, i.e., $\tau_c(z) = z, \forall c \in \{1, \ldots, C\}$;
- CLASSEMES-BIT We create a binary vector by thresholding the values at zero, i.e., we choose $\tau_c(z) = \mathbf{1}[z > 0], \forall c \in \{1, \ldots, C\}$ where $\mathbf{1}[.]$ is the 0-1 indicator function of its boolean argument.

3.3 PiCoDes

As we will show in the experiments, the training procedure presented in Sec. 3.2 can efficiently learn classifier-based descriptors yielding state-of-the-art accuracy. However, it also comes with a few shortcomings:

- The basis classifiers are learned disjointly by optimizing an objective that is unrelated to their actual usage as features at application time (see Eq. 4).
- The basis classes are hand-selected real object classes. The choice of classes to use is left to the designer of the descriptor but there is no well-defined criterion to determine the best set of categories.
- The quantization of the descriptor is applied as a post-processing and not considered in the learning objective of the basis classifiers.

We now describe a method that addresses these limitations. The idea of the approach is quite simple: given that we want to use basis classifiers as features with linear models, we design our learning objective to enforce that linear combinations of these basis classifiers must yield good accuracy with respect to our training set. In other words, rather than hand-designing the basis classes, we learn abstract categories aimed at optimizing linear classification when they are used as features. This learning objective decouples the number of training classes from the target dimensionality of the binary descriptor and thus it allows us to optimize our descriptor for any arbitrary length. Furthermore, it enables direct optimization of the learning parameters with respect to the desired quantization function τ_c, avoiding the suboptimal use of quantizers applied as a post-processing.

Figure 1 provides a visualization of a few basis classifiers learned by our training procedure. This figure suggests that our learned features describe the image in terms of binary visually-interpretable properties corresponding, e.g., to particular shape, texture or color patterns.

We first introduced this approach in [18]. We called these features PICODES, which stands for "Picture Codes" but also "Pico-Descriptors". In order to obtain a highly compact descriptor, we optimize the PICODES features to be *binary*, i.e., $h(\boldsymbol{x}; \boldsymbol{a}_c) = \mathbf{1}[\boldsymbol{a}_c^T \boldsymbol{x} > 0]$ (note that this is the same form as in Eq. 1 with only difference that here we choose $\tau_c(z) = \mathbf{1}[z > 0]$).

3.3.1 Learning the basis classifiers

In order to learn the PICODES basis classifiers, we use a binary encoding of the labels for the training examples in $\mathcal{D}^S$: for each example i, we define $\boldsymbol{y}_i \in \{-1, +1\}^K$ to be a vector describing its category label, with $y_{ik} = +1$ iff the i-th example belongs to class k. As seen shortly, this labeling will allow us to implement a form of "one-vs-the-rest" strategy for training the basis classifiers.

We want to optimize the parameter matrix $\boldsymbol{A}$ so that linear combinations of the resulting basis classifiers yield good categorization accuracy on $\mathcal{D}^S$. In order to formalize this objective, for each training class k we introduce auxiliary parameters $(\boldsymbol{w}_k, b_k)$, which define a linear classifier operating on the PICODES features and distinguishing the k-th class from all the others. Thus, we state our overall objective as *joint* optimization over our representation (parameterized by $\boldsymbol{A}$) and the auxiliary parameters defining the K linear classifiers so as to obtain the best possible linear classification accuracy on $\mathcal{D}^S$. Specifically, we use a large-margin formulation which trades 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:

$$E(\boldsymbol{A}, \boldsymbol{w}_{1..K}, b_{1..K}) = \sum_{k=1}^{K} \left\{ \frac{1}{2} \|\boldsymbol{w}_k\|^2 + \frac{\lambda}{N} \sum_{i=1}^{N} \ell \left[y_{i,k} (b_k + \boldsymbol{w}_k^\top \boldsymbol{h}(\boldsymbol{x}_i; \boldsymbol{A})) \right] \right\} \quad (6)$$

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

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

We propose to jointly learn the linear SVMs $(\boldsymbol{w}_k, b_k)$ and the parameters $\boldsymbol{a}_c$ of the basis classifiers using the method described below.

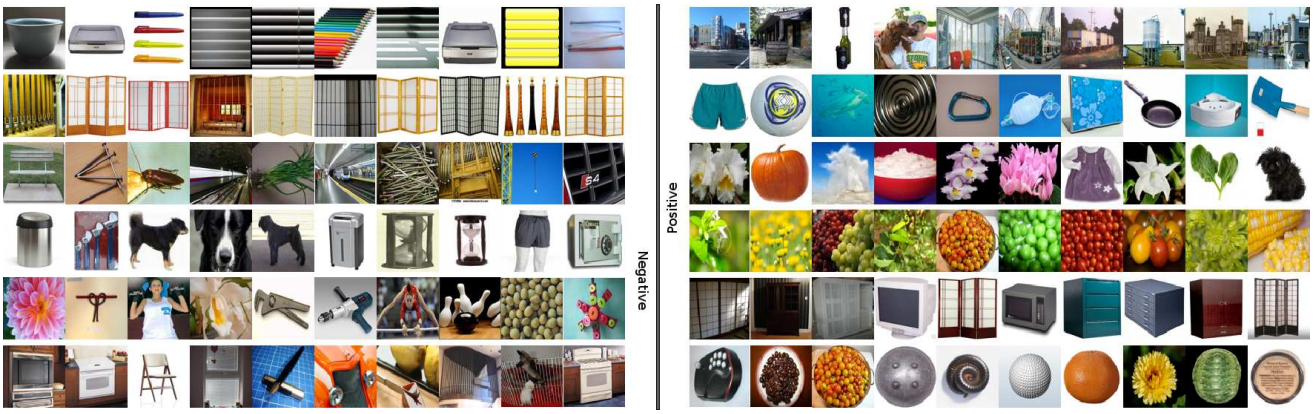

Fig. 1. Visualization of PICODES. Each row of images illustrates a particular bit in our 128-bit descriptor. These 6 randomly chosen bits are shown as follows: for bit c, all images are sorted by non-binarized classifier outputs $\boldsymbol{a}_c^\top \boldsymbol{x}$ and the 10 smallest and largest are 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.

3.3.2 Optimization

Minimizing this objective requires optimization directly for binary features, which is difficult and non-convex. Thus, we use an alternation scheme implementing 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 PICODES

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 (see the supplementary material) that in this case the objective becomes:

$$E(\boldsymbol{a}_c) = \sum_{i=1}^{N} v_i \mathbf{1}[z_i \boldsymbol{a}_c^T \boldsymbol{x}_i > 0] + const \qquad (7)$$

where $z_i \in \{-1, +1\}$ and $v_i \in \mathbb{R}^+$ are known values computed from the fixed parameters. Unfortunately, this objective is not convex and not trivial to optimize. Thus, we replace it with the following convex upper bound defined in terms of the hinge function ℓ:

$$\hat{E}(\boldsymbol{a}_c) = \sum_{i=1}^{N} v_i \ell(z_i \boldsymbol{a}_c^T \boldsymbol{x}_i) \qquad (8)$$

This objective can be globally optimized using an LP solver or software for SVM training. We had success with LIBLINEAR [22], which deals nicely with the large problem sizes of both our optimization steps.

We have also experimented with several other optimization methods, including stochastic gradient descent applied to a modified version of our objective, but they led to inferior results. Please refer to the supplementary material for further details.

3.4 Meta-Classes

We have seen that the method to learn CLASSEMES introduced in Sec. 3.2 is very simple to implement and scalable to the size of the training database but it is sub-optimal in several aspects. Conversely, the PICODES approach described in Sec. 3.3 provides principled learning of abstract basis classes explicitly optimized for good accuracy with linear models, but it requires a computationally expensive minimization, which in practice can be run only for very compact dimensionalities (our largest PiCoDes contain 2048 bits and required several weeks to be learned). We now introduce a descriptor-learning algorithm that combines the advantages of the previous two methods: *1)* scalable learning; *2)* automatic discovery of abstract basis-classes yielding good recognition when used as features with linear models. We originally presented this approach in [19].

We refer to the basis classes learned by this method as "meta-classes". Intuitively, we want our meta-class classifiers to be "repeatable" (i.e., they should produce similar outputs on images of the same object category) and to capture properties of the image that are useful for categorization. We formalize this intuition by defining each meta-class to be a subset of object classes in the training set. Specifically, we hierarchically partition the set of training object classes such that each meta-class subset can be easily recognized from the others. This criterion forces the classifiers trained on the meta-classes to be repeatable. At the same time, since the meta-classes are superclasses of the original training categories, by definition the classifiers trained on them will capture common visual properties shared by similar classes while being effective to discriminate visually-dissimilar object classes.

3.4.1 Discovering the meta-classes

In this section we describe the procedure to discover the meta-classes. Our method is an instance of the

algorithm for label tree learning described in [23]. (Note that other label-tree learning methods, such as [24], [25], could also be applied to our task of meta-class training.) This algorithm learns a tree-structure of classifiers (the label tree) and was proposed to speed up categorization in settings where the number of classes is very large. Instead, here we use the label tree training procedure to learn meta-classes, i.e, sets of classes that can be easily recognized from others. We provide below a review of the label tree algorithm, contextualized for our objective.

Let $\ell_{\mathcal{D}}$ be the set of distinct class labels in the training set $\mathcal{D}^S$, i.e. $\ell_{\mathcal{D}} \equiv \{1, \ldots, K\}$. The label tree is generated in a top-down fashion starting from the root of the tree. Each node has associated a set of object class labels. The label set of the root node is set equal to $\ell_{\mathcal{D}}$. Let us now consider a node with label set ℓ. We now describe how to generate its two children. (Although the label tree can have arbitrary branching factor at each node, in our work we use binary trees.) The two children define a partition of the label set of the parent: if we denote with ℓ^L and ℓ^R the label sets of the two children, then we want $\ell^L \cup \ell^R = \ell$ and $\ell^L \cap \ell^R = \emptyset$. Ideally, we want to choose the partition $\{\ell^L, \ell^R\}$ so that a binary classifier $h_{(\ell^L,\ell^R)}(\boldsymbol{x})$ trained to distinguish these two meta-classes makes as few mistakes as possible. In order to select the best classifier, we should train a classifier for each of the possible $(|\ell|(|\ell|-1)/2-1)$ partitions of ℓ, but this operation is prohibitively expensive. Instead, we take inspiration from the work of [23] and we use the confusion matrix of one-vs-the-rest classifiers learned for the individual object classes to determine a good partition of ℓ: intuitively, our goal is to include classes that tend to be confused with each other in the same label subset. More formally, let $\hat{h}_1, \ldots, \hat{h}_{|\ell_{\mathcal{D}}|}$ be the one-vs-the-rest LP-β classifiers learned for the individual object classes using the training set $\mathcal{D}^S$. Let $A \in \mathbb{R}^{|\ell_{\mathcal{D}}| \times |\ell_{\mathcal{D}}|}$ be the confusion matrix of these classifiers evaluated on a separate validation set $\mathcal{D}^{\text{val}} \subset \mathcal{D}^S$: A_{ij} gives the number of samples of class i in $\mathcal{D}^{\text{val}}$ that have been predicted to belong to class j (we assume the winner-take-all strategy for multiclass classification). Since this matrix is not symmetric in general, we compute its symmetrized version as $B = (A + A^T)/2$. Then, for each node we propose to partition its label set ℓ into the subsets $\ell^L \subset \ell$, $\ell^R \equiv \ell - \ell^L$ that maximize the following objective:

$$E(\ell) = \sum_{i,j \in \ell^L} B_{ij} + \sum_{p,q \in \ell^R} B_{pq} \ . \tag{9}$$

The objective encourages to include in the same subset classes that are difficult to tell apart, thus favoring the creation of meta-classes containing common visual properties. At the same time, maximizing this objective will tend to produce meta-classes ℓ^L, ℓ^R that are easy to separate from each other.

Note that optimization of eq. 9 can be formulated as a graph partitioning problem [26]. We compute the solution ℓ^L by applying spectral clustering [27] to the matrix B: this is equivalent to solving a relaxed, normalized version of eq. 9 that penalizes unbalanced partitions. We repeat this process recursively on each node until it contains a single class label, i.e., $|\ell| = 1$.

Fig. 2 shows a portion of our meta-class tree learned from the *ImageNet* dataset [1]. As expected, we found that our meta-classes tend to group together object classes that are visually similar although not necessarily semantically related (e.g., lantern and electric lamp but also hurricane lamp and perfume).

3.4.2 Learning the meta-class classifiers

The learning of the basis classifiers is now straightforward: at each node ℓ of the label tree we train an LP-β classifier on the binary split $\{\ell^L, \ell^R\}$. Specifically, let I^ℓ denote the indices of training examples having class labels in ℓ. Then, we form the labeled set $\mathcal{D}_{(\ell^L,\ell^R)} = \{(\boldsymbol{x}_i, +1) : i \in I^{\ell^L}\} \cup \{(\boldsymbol{x}_i, -1) : i \in I^{\ell^R}\}$ and use it to train meta-class classifier $h_{(\ell^L,\ell^R)}(\boldsymbol{x})$. This procedure yields in total $\tilde{C}$ basis classifiers, where $\tilde{C}$ is the number of inner nodes of the label tree. Note that this learning is embarrassingly parallelizable as we can learn the hypothesis independently. We also include in the descriptor the outputs of the one-vs-the-rest classifiers $\hat{h}_1, \ldots, \hat{h}_{|\ell_{\mathcal{D}}|}$, as we have found that this improves the final performance of the classifier. The image descriptor $\boldsymbol{h}(\boldsymbol{x})$ is then a C-dimensional vector, where $C = \tilde{C} + K$.

We experiment with two versions of the function τ in Eq. 1, thus giving rise to the two following variants of the meta-class descriptor:

- MC: We convert the raw score of each LP-β classifier into a probabilistic output by means of a sigmoid function, i.e., we set $\tau_c(z) = 1/(1+\exp(-\alpha_c z + \beta_c))$. We learn the parameters of the sigmoid by means of Platt's scaling procedure [28], using the validation set $\mathcal{D}^{\text{val}}$. We found that this sigmoidal normalization yields a large boost in the accuracy of the final classifier trained on this representation, probably as it makes the range of classification scores more homogeneous and reduces outlier values.
- MC-BIT: We create a binary vector by setting set $\tau_c(z) = \mathbf{1}[z > 0], \forall c \in \{1, \ldots, C\}$.

3.5 Encoding local information

The framework described in Sec. 3.1 describes methods to compute *global* classifier-based descriptors, i.e., feature vectors where each individual entry is the output of a classifier evaluated over the *entire* image. While our experiments show that these representations produce high accuracy on the task of whole-image classification, they are clearly not suitable for recognition when the object of interest occupies only a

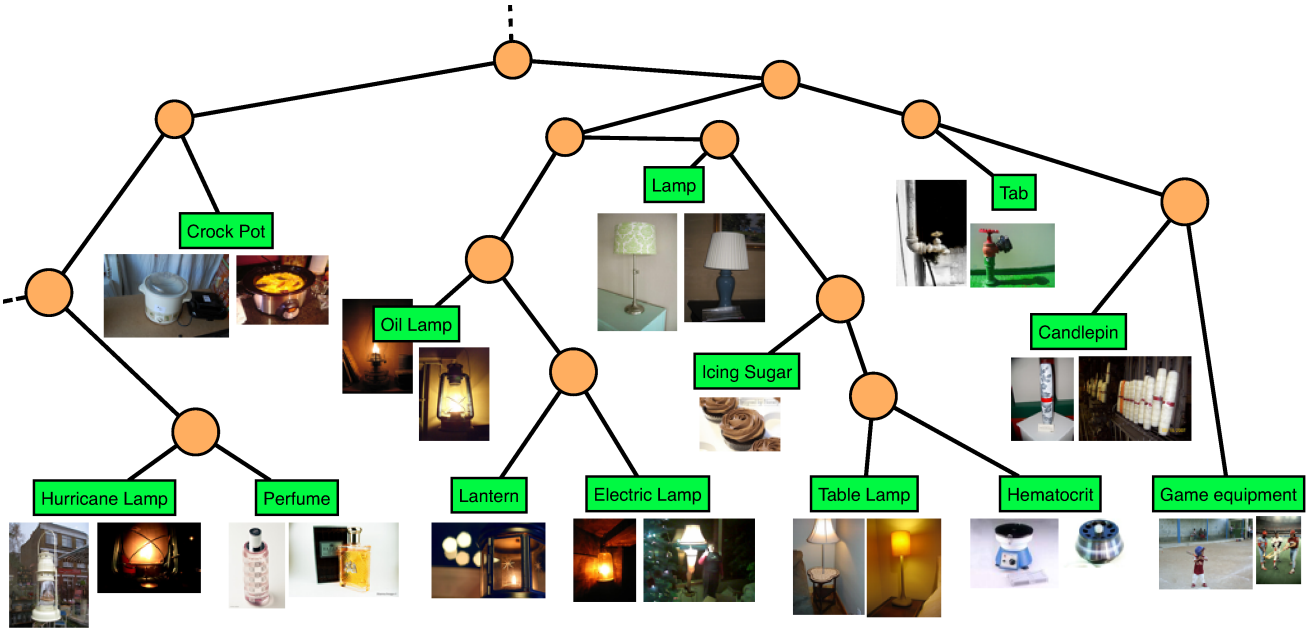

Fig. 2. *Meta-class tree.* This figure shows a small portion of the tree learned by the *meta-class* approach described in Sec 3.4. The rectangular nodes represent the leaves of the tree, associated with real categories of the *ImageNet* dataset. The inner round nodes represent the meta-classes automatically learned by our method, which tends to group together categories that are difficult to tell apart.

small region of the photo. In this section we describe how to extend our framework to encode local information in the descriptor so as to handle cases when the image contains small or multiple objects.

We present three different strategies to capture local information. Each of these local encoding methods can be applied to any of the descriptors described in Sec. 3.2, 3.3, and 3.4. At a high-level, each local encoding method is defined by a splitting function s that decomposes the image into a set of M subimages, and a combiner m that aggregates the M vectors produced by extracting our classifier-based descriptor $\boldsymbol{h}$ from the individual subimages. More formally, let $\mathcal{X}$ be the space of all images. Thus, the splitting function $s : \mathcal{X} \rightarrow \mathcal{X}^M$ takes an image $\boldsymbol{x} \in \mathcal{X}$ as input and outputs M rectangular subimages $\{\boldsymbol{x}^{(m)}\}_{m=1}^{M}$. Then, we define our final image descriptor $\bar{\boldsymbol{h}}(\boldsymbol{x})$ capturing local information as $\bar{\boldsymbol{h}}(\boldsymbol{x}) = m(\boldsymbol{h}(\boldsymbol{x}^{(1)}), \ldots, \boldsymbol{h}(\boldsymbol{x}^{(M)}))$. In the next subsections we describe several choices of functions s and m, giving rise to different $\bar{\boldsymbol{h}}(\boldsymbol{x})$.

3.5.1 SPCAT: Spatial Pyramid + Concatenation

This is an instantiation of the *spatial pyramid* method proposed in [29] where the function s decomposes the image into a hierarchical partition of rectangular subimages. The pyramid consists of L layers, where the first layer (layer 0) is the image itself. Each layer spatially subdivides each subimage of the previous layer into a grid of 2x2 subimages of equal size. Hence the total number of rectangular subimages defined by the pyramid is $M = \sum_{l=0}^{L-1} 4^l$.

The combiner m takes as input the set of descriptors computed from the individual subimages of the pyramid, and concatenates them together:

$$\bar{\boldsymbol{h}}(\boldsymbol{x}) = \left[\boldsymbol{h}(\boldsymbol{x}^{(1)})^\top, \ldots, \boldsymbol{h}(\boldsymbol{x}^{(M)})^\top\right]^\top .$$

The final descriptor $\bar{\boldsymbol{h}}$ has dimensionality $C \cdot M$, where C is the length of the individual descriptors associated to the regions (as defined in Sec. 3.1).

3.5.2 SPLPOOL: Spatial Pyramid + Layer Pooling

In this strategy the function s still implements a Spatial Pyramid. Instead, the combiner function m now concatenates descriptors obtained by pooling (or aggregating) the feature vectors within each layer. Specifically, let $\boldsymbol{x}^{(l,g)}$ be the g-th subimage of the l-th layer. In case of *binary* descriptors (CLASSEMES-BIT, PICODES, and MC-BIT) we keep the final features binary by performing max-pooling within each layer:

$$\bar{\boldsymbol{h}}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{h}(\boldsymbol{x}) \\ \max_{g=1,\ldots,4} \boldsymbol{h}(\boldsymbol{x}^{(1,g)}) \\ \vdots \\ \max_{g=1,\ldots,4^{L-1}} \boldsymbol{h}(\boldsymbol{x}^{(L-1,g)}) \end{bmatrix}$$

where the function $\max$ computes the component-wise maximum. In case of *real-valued* descriptors (CLASSEMES and MC), instead we average the feature values within each layer (thus replacing $\max$ with the sample mean of each feature). We also tried max-pooling for the real-valued descriptor but we found empirically that this yields lower accuracy. With this local encoding strategy, the final vector $\bar{\boldsymbol{h}}(\boldsymbol{x})$ has dimensionality $L \cdot C$.

3.5.3 OBJPOOL: Objectness + Pooling

The encoding methods of subsections 3.5.1 and 3.5.2 are limited by the fact that the subdivision of the image is fixed, and does not take into account the actual locations of the objects in the photo. Intuitively, we would like the function s to split the image into regions containing objects so as to encode more relevant subimages. To this end, we propose to implement s as a class-generic object detector producing a candidate set of regions that are likely to contain an object. For this purpose, we define the function s to return the M rectangular subimages $\{\boldsymbol{x}^{(m)}\}_{m=1}^{M}$ that have the greatest *ObjectNess* measure [30]. Then, the combiner m performs average pooling, i.e., computes the average of each feature entry over all M subimages. Even in this case we tried max-pooling, but again we found this strategy to produce inferior results compared to average pooling. We append the resulting descriptor to the feature vector computed from the entire image:

$$\bar{\boldsymbol{h}}(\boldsymbol{x}) = \left[\boldsymbol{h}(\boldsymbol{x})^\top, \quad \frac{1}{M}\sum_{m=1}^{M}\boldsymbol{h}(\boldsymbol{x}^{(m)})^\top\right]^\top .$$

Thus, the dimensionality of the final descriptor is in this case $2 \cdot C$.

4 Experiments

4.1 Datasets

In this work we make use of several datasets, for both learning and testing our descriptors. Here we summarize briefly their characteristics:

- Caltech 256 [3]: Popular benchmark for object categorization involving $\sim$ 30K images partitioned into 256 visual categories. Each photo contains a single centered object.
- ImageNet [1] (Spring 2010 release): Large-scale image dataset consisting of more than 15K object categories and 11M pictures. Most images contain a single object.
- ILSVRC2010 [4]: Large-scale benchmark for object categorization. It is a subset of the ImageNet dataset: it contains 1000 categories and 1.2M images.
- PASCAL VOC 2007 [31]: Benchmark for object detection and classification including 20 object categories and 9,963 images. Despite the small size of this dataset, the classification is very challenging as the scale and position of the objects vary greatly.
- MIT 67 [32]: Benchmark for indoor scene recognition. It consists of 67 indoor scenes (e.g. corridors, bookstores) for a total of 15,620 images, each usually containing multiple objects.
- SUN 397 [33]: Large-scale benchmark for indoor/outdoor scene recognition. It contains 397 scene categories for a total number of 108,754 images.

4.2 Low-level descriptors

As previously described in Sec. 3.1, our descriptors are built upon a set of low-level features. For the features that are based on the aggregation of local descriptors, we also exploit the Spatial Pyramid method [29] to encode weak geometry into the representation. In particular we use a pyramid (SP) of L layers, and for each layer $l \in \{0, \ldots, L-1\}$ we partition the image into a $2^l \times 2^l$ grid of cells and we extract a low-level feature vector from each cell. We use the following low-level features:

- *Color GIST* [21]: we first resized the images to 32×32 pixels, without maintaining the aspect ratio, and then we compute the orientation histograms on a 4×4 grid. We use 3 scales with the number or orientation per scale being 8, 8, 4.
- *Oriented HOG* [34] (4 SP layers): Histogram of Oriented Gradients computed using 20 bins.
- *Unoriented HOG* [34] (4 SP layers): an histogram of unoriented gradients quantized into 40 bins.
- *SSIM* [35] (3 SP layers): we compute a histogram by extracting the 30-dimensional self-similarity descriptor every 5 pixels, and by quantizing it into 300 cluster centroids obtained from K-means.
- *SIFT* [20] (3 SP layers): The 128-dimensional SIFT descriptors are computed from the interest points returned by a DoG detector [36]. We finally compute a Bag-Of-Word histogram of these descriptors, using a K-means vocabulary of 500 words.

In our work we treat each pyramid layer as if it was a separate low-level feature vector. The concatenation of all these $M = 15$ feature vectors, yields a $22,860$-dimensional vector. As described in Sec. 3.1 we make use of the explicit map proposed in [12] to perform efficient non-linear classification with an approximated Intersection Kernel. The map is a function $\Psi(\boldsymbol{f}; r, L, \gamma)$ that takes a feature vector $\boldsymbol{f}$ as input and is parameterized by r (number of samples), L (period of the sampling), and γ (normalization of the kernel). We set $r = 1$ for all the features, producing feature vectors three times as big as the original vectors, and $\gamma = 1$ as suggested in [12]. For each feature vector, we select the parameter L by grid search, minimizing the error between the exact kernel distance K, and the approximated one, i.e. $\min_L \sum_{i,j=1}^{N} \left|\left\langle \Psi(\boldsymbol{f}_i), \Psi(\boldsymbol{f}_j)\right\rangle - K(\boldsymbol{f}_i, \boldsymbol{f}_j)\right|$ computed on a validation set consisting of $N = 2560$ images randomly sampled from the Caltech 256 dataset [37]. The concatenation of all these mapped low-level feature vectors form the vector Ψ introduced in Eq. 2, which for this implementation has dimensionality $D = 68,580$. In this section we will refer to this descriptor as PSI. Finally, we want to emphasize that our approach is obviously not constrained to work only with the particular choice of low-level features described above. Therefore, it could easily take advantage of more powerful low-level features,

Name	Dimens.	Storage size per image
Our descriptors		
PSI (Sec. 4.2)	68580 (real)	268 KB
CLASSEMES (Sec. 3.2)	2659 (real)	11 KB
CLASSEMES-BIT (Sec. 3.2)	2659 (bin)	0.33 KB
PICODES (Sec. 3.3)	2048 (bin)	0.25 KB
MC (Sec. 3.4)	15232 (real)	60 KB
MC-BIT (Sec. 3.4)	15232 (bin)	1.9 KB
MC-LSH (Sec. 3.4)	200 K (bin)	25 KB
X+SPCAT L0L1 (Sec. 3.5.1)	$5\times$	$5\times$
X+SPLPOOL L0L1 (Sec. 3.5.2)	$2\times$	$2\times$
X+OBJPOOL (Sec. 3.5.3)	$2\times$	$2\times$
Prior work		
Gong et al. 2011, [38] (ITQ)	2048 (bin)	0.25 KB
Gong et al. 2011 [38] (CCA-ITQ)	2048 (bin)	0.25 KB
Li et al. 2010 [17] (OBJECTBANK)	44604	175 KB

TABLE 1
Descriptors considered in our comparison. This table presents: the name of the descriptor and the section where it is introduced; the native dimensionality and whether or not the descriptor is binary; the required memory to store a single image descriptor.

such as the recently introduced Fisher Vectors [2], which have been shown to lead to state-of-the-art results in object categorization.

4.3 Learning classifier-based descriptors

In this section we describe in detail how we implemented the classifier-based descriptors proposed in Sec. 3. A summary of the descriptors described in this section is provided in Table 1.

4.3.1 Classemes

The training dataset $\mathcal{D}^S$ for Classemes is obtained from concepts in the Large Scale Concept Ontology for Multimedia (LSCOM) [39], which includes textual descriptions of a collection of concepts selected to be useful, observable and feasible for automatic visual detection, and as such are likely to form a good basis for image retrieval and object recognition tasks.

For each of the selected 2659 categories, the top 150 images were gathered from the *bing.com* image search engine, using the LSCOM textual descriptions as queries. These pictures form the training set $\mathcal{D}^S$ that in total has $\sim$400K images. We finally learned the classeme descriptors CLASSEMES and CLASSEMES-BIT, using the procedure described in Sec. 3.2. Since the basis classifiers can be learned independently, we exploited a cluster of machines to reduce the training time.

Note that the textual descriptions of 28 categories overlap with the name of some categories of the popular benchmark Caltech 256 [40], which is used as one of the test datasets in Sec. 4.5.1. This "pre-training" of the query classes is in practice not significant: removing the corresponding 28 values from the descriptor, causes a negligible drop in performance on Caltech 256, $< 1\%$.

4.3.2 PiCoDes

We built the training set $\mathcal{D}^S$ from 2659 randomly selected ImageNet synsets using 30 images per category, for a total of $\sim$ 80K images. In order to avoid pre-learning the test classes in the descriptor, we avoided picking as training synsets categories belonging to the ILSVRC2010 dataset or related to Caltech 256 (we performed sub-string matching comparison between the synset tags and the Caltech 256 keywords, removing in total 711 ImageNet classes). This allows us to evaluate our descriptor in a scenario where each test class to recognize is effectively novel, i.e., not present in the training set used to learn the descriptor.

Note that the learning procedure described in Sec. 3.3 is not easily parallelizable, and it requires continuous access to the high-dimensional image descriptors $\Psi(\boldsymbol{x})$ of Eq. 2, which are very costly in terms of storage (see PSI in Tab. 1). Thus, in practice, we compress down the vectors via PCA, producing a vector of 6415 dimensions. More formally, for the PICODES learning, we set $\Psi(\boldsymbol{x}) \equiv \mathbf{P}\Psi(\boldsymbol{x})$ where $\mathbf{P}$ contains the top 6415 PCA components. The dimensionality of 6415 was chosen based on a preliminary experiment of multiclass classification on Caltech 256 (please refer to the supplementary material).

Given the training set $\mathcal{D}^S$ we can train the PiCoDes descriptor as described in Sec. 3.3, performing 15 iterations. For each target dimensionality C, we learned multiple descriptors using different values for the hyper-parameter λ, and kept the descriptor that gave the lowest multiclass error on a validation set consisting of 5 images per class. In this paper we report results for $C = 2048$.

4.3.3 Meta-Classes

We formed the training set $\mathcal{D}^S$ from 8000 randomly sampled synsets, using at most 1000 examples per category. We also created a validation set $\mathcal{D}^{\text{val}}$ with the same categories and 80 examples per class. As done for PiCoDes, we selected the training classes such that the synsets of these categories do not contain any of the Caltech 256 or ILSVRC2010 class labels, so as to avoid "pre-learning" the test classes during the feature-training stage. We learned the MC and MC-BIT descriptors following the procedure described in Sec. 3.4, using the validation set to compute the confusion matrix and the two parameters of the sigmoid function for each meta-class. Since the basis classifiers can be learned independently, we use a cluster of multiple machines to reduce the training time.

4.4 Evaluation setup

The next sections will describe the experimental evaluations that we have done using our descriptors, as well as other popular image representations. All the experiments involve the usage of a classifier to predict the correct object/scene label of an image,

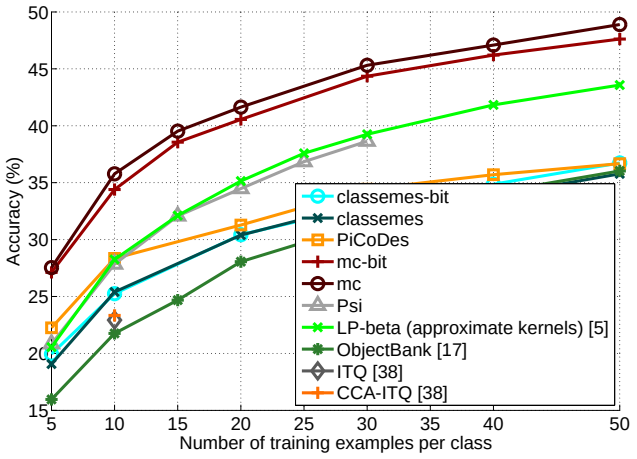

Fig. 3. Multi-class recognition on Caltech 256 using different image representations. The classification model is a linear SVM (except for LP-β). The accuracy is plotted as a function of the training set size.

and we decided to use the very efficient linear SVM model (with the only exception of LP-β). In cases of multi-class categorization, we used the 1-vs-the-rest strategy and performed the prediction using the winner-take-all strategy. The SVM hyperparameter is selected using either 5-fold cross validation or using the validation set when available.

4.5 Experiments: Object Class Recognition

4.5.1 Caltech 256

We present experiments obtained with several image descriptors on the challenging Caltech 256 benchmark. We follow the standard approach of learning the classifier for different number of training examples per class $\{5, 10, \ldots, 50\}$. We evaluate the model on a test set consisting of 25 examples per class, and report the accuracy as the mean of the diagonal of the confusion matrix. We compare our descriptors CLASSEMES, CLASSEMES-BIT, PICODES, MC and MC-BIT with the following methods:

- PSI: the concatenation of the mapped low-level features (introduced in Sec. 4.2). This baseline is interesting to consider as it shows the accuracy that can be obtained by directly training the final classifier on the low-level image representation that we have used to learn our descriptors.
- ITQ: the embedding method introduced in [38], which learns a binary code by directly minimizing the quantization error of mapping the input data to vertices of the binary hypercube. As training data for the learning we use 2560 images from the Caltech 256 dataset: the samples are converted into PSI descriptors by using the same low-level features, feature mapping, and PCA projection that we have adopted for PiCoDes. We have tried different PCA subspace dimensionality; here we report the results obtained with the setting yielding the best final accuracy. We learn a ITQ descriptor of 2048 bits in order to compare it with PICODES.
- CCA-ITQ. Same as *ITQ* but instead of PCA we use Canonical Correlation Analysis (CCA) (as suggested in [38]), which performs discriminative dimensionality reduction.
- OBJECTBANK: the descriptor introduced in [17] which encodes into a single vector both semantic and spatial information of objects detected in the input image, using a set of pre-trained detectors.
- LP-β : this denotes the variant of the LP-β [5] multiple-kernel combiner described in Sec. 3.2, based on the same low-level features used by classemes and PiCodes (see Sec. 4.2).

Figure 3 shows the multi-class recognition accuracy of the different approaches as a function of the number of training examples per class. Note that the differences in performance between CLASSEMES and CLASSEMES-BIT is negligible on this benchmark, probably indicating that the accuracy of the classifier is saturated at this dimensionality and cannot exploit the additional information provided by the continuous data. OBJECTBANK performs moderately well but much worse than our descriptors. PICODES outperforms CLASSEMES as the former are explicitly optimized for linear classification, which is the actual usage of the descriptors in this test. For the same dimensionality and storage cost, PICODES outperform also ITQ and CCA-ITQ. We observed that for very small descriptor dimensionalities, CCA-ITQ performs slightly better than PICODES (e.g., by a margin of 2% for 128 bits). Note however that in our tests the CCA transformation was given the advantage of using the Caltech 256 classes as training data.

The methods PSI and LP-β use multiple features and nonlinear kernels (albeit in approximate form) and hence their recognition accuracies are among the best in the literature. However, for small numbers of training examples per class, PICODES surprisingly matches LP-β . Furthermore, note from Table 1 that the storage cost is 2 order of magnitude higher than PICODES.

Finally, we can see that our descriptors MC and MC-BIT greatly outperform all the other representations; note that the binarized version (MC-BIT) is only 1% worse than the real-valued descriptor (MC) while being 32 times more compact (the storage size for MC-BIT is less than 2KB per image). Moreover we use a simple linear model, which enables efficient training and recognition, and the storage requirement for these descriptors is only a few KBytes per image.

4.5.2 ILSVRC 2010

We now present results of multiclass recognition on the ILSVRC 2010 dataset. The large size of this database poses new challenges and issues that are not present in smaller databases, as already noted in [41],

Method	Top-1 accuracy (%)	Storage 10M images (GB)	Recognition time (μs)
CLASSEMES-BIT	22.15	3.09	6.67
CLASSEMES	29.95	99.05	1.53
PICODES	22.66	2.38	5.14
MC-BIT	36.71	18	38.23
MC-LSH	42.15	232	501.97
Lin et al. 2011 [9]	52.9	43,945	796.60
Sanchez et al. 2011 [2] (FV)	n/a	39,041	603.1
Sanchez et al. 2011 [2] (FV+PQ)	54.3	1,220	2131.0

TABLE 2
Object class recognition on ILSVRC 2010. For each descriptor, we report: the top-1 accuracy on the test set; the storage size of an hypothetical database consisting of 10 million images; the average time required to evaluate a linear classifier on a single image, without including the feature-extraction time (see supplementary material for details). All the methods use linear SVM as classifier.

[2], [9]. Yet, our binary CLASSEMES-BIT, PICODES, and MC-BIT features render the learning on this database relatively fast to accomplish even on a budget PC, as we can represent these descriptors using a single bit per dimension, storing the entire ILSVRC2010 training set in at most 2.13 GB of memory. This allows us to use efficient software for batch training of linear SVM: we have had good success with LIBLINEAR [22], by simply modifying the code to support input data in bitmap format. To further speed-up the learning we reduce the negative training set by sampling $n = 150$ examples per class. According to our study, this sampling causes only a negligible drop in accuracy. Training a single one-vs-the-rest linear SVM using our MC-BIT descriptor takes on average 50 seconds. So the entire multiclass training for ILSVRC2010 can be accomplished in 14 hours on a single-core computer. In practice the learning can be made highly parallel when multiple cores and machines are available. As a comparison, the winning system of the ILSVRC2010 challenge [9] required a week of training with a powerful cluster of machines and specialized hardware.

In table 2 we show the results of the analysis we made with our descriptors and other approaches reported in the literature on the ILSVRC2010 dataset. Note that the MC-LSH method is just a compressed form of MC, obtained using LSH with $200K$ random projections. Our experiments suggest that this compression yields no significant degradation in accuracy, while reducing considerably the storage requirement. We can notice that the performances of the descriptors CLASSEMES and MC-LSH are remarkably superior to the binary versions CLASSEMES-BIT and MC-BIT, yielding an improvement of +7.8% and +5.44%, respectively. This indicates that the real-valued descriptors are more informative than the corresponding binary versions, but this additional expressiveness is exhibited only in large-scale scenarios (again, we remind the reader that on the small Caltech 256 dataset, no significant improvement was obtained using real-valued descriptors). In particular MC-LSH achieves a recognition rate of 42.15% and a top-5 accuracy of 64.01% (top-5 accuracy is the traditonal performance measure of ILSVRC 2010). The systems of [2] and *NEC* are based on very high-dimensional image signatures and linear classifiers. Note that although these approaches provides better raw accuracy, storage requirements and prediction times are orders of magnitude more costly and clearly inapplicable in our motivational large scale scenarios (see a discussion in Sec. 4.7).

We also provide results achieved with the individual subcomponents of MC-BIT: "MC-BIT-TREE", which consists of the 7232 meta-class classifiers learned for the inner nodes of the label tree, yields 33.87% of Top-1 accuracy; "MC-BIT-1VSALL", which contains the outputs of the 8000 one-vs-the-rest classeme classifiers yields 30.64%. Note that the accuracy obtained using only the label tree features is clearly superior to the one generated by only the classeme features. This indicates that the grouping of classes performed by the label tree learning produces features that lead to better generalization on novel classes. However, the complete MC-BIT descriptor yields even higher accuracy (36.71%), suggesting that there is value in using both subcomponents. Please refer to the supplementary material for additional information.

A natural question is: how do our learned meta-classes compare to the semantic nodes in the hand-constructed ImageNet hierarchy? For example, [16] showed that this hierarchy provides a form of prior semantic knowledge that can be effectively exploited to define a metric for retrieval. In order to run a fair comparison with our method, we pruned the original ImageNet tree so as to leave only the nodes corresponding to the 8,000 synsets used to train our MC descriptor. This produced a new semantic tree with 1,528 internal nodes. As before, we then trained a binary classifier for each of these semantic classes and obtained a new binary descriptor of 1,528 features. We found that these "semantic meta-classes" yield an accuracy of 18.37% on ILSVRC 2010. However, note that a random selection of 1,528 features from our MC-BIT descriptor performs much better, yielding an average accuracy of 21.14% (the accuracy is averaged over 10 random selections of 1,528 features). This suggests that meta-class features automatically learned by considering visual relations between classes are more effective than attributes based on human-defined notions of semantic similarity. Further comparative results are given in the supplementary material.

4.5.3 PASCAL 2007

We now present categorization results on PASCAL 2007 using our descriptors. To the best of our knowledge, this is the first comprehensive empirical

Method	mAP
CLASSEMES-BIT	0.427
CLASSEMES-BIT + SPLPOOL L0L1	0.452
CLASSEMES	0.438
CLASSEMES + SPLPOOL L0L1	0.447
PICODES	0.437
PICODES + SPLPOOL L0L1	0.455
MC-BIT	0.527
MC-BIT + SPLPOOL L0L1	0.53
MC	0.532
MC + OBJPOOL	0.55
Li et al. 2010 [17] (OBJECTBANK)	0.452
Harzallah et al. 2009 [42]	0.635
Song et al. 2011 [43]	0.705

TABLE 3
Object-class categorization results obtained on PASCAL 2007 using our descriptors and other methods in the literature. The performance measure is the mean of the Average Precision. For our descriptors, the classification model is a linear SVM.

study of the recognition accuracy of classifier-based image descriptors on a *detection* dataset: this benchmark is quite challenging as each image contains multiple object whose position and scale varies greatly. For this reason we tested our descriptors using the local-encoding extensions described in Sec. 3.5. We implemented the method SPLPOOL L0L1 using a pyramid consisting of two levels, and the extension OBJPOOL using the 25 subwindows with the greatest ObjectNess score. Note that we tried additional pyramid levels as well as increasing the number of ObjectNess-subwindows, but we did not see a significant improvement in accuracy. Table 3 summarizes the results of our experiments in terms of mAP. Despite the simplicity of the linear classification model that we are using, we can see that our descriptors yield good accuracy while enabling extremely efficient prediction. Moreover note that all the proposed local-encoding strategies boost the accuracies of the raw descriptors. In particular OBJPOOL produces the best results while only doubling the storage cost.

4.6 Experiments on Scene Recognition

4.6.1 MIT 67

We present in this section the results of the experiments performed on the MIT 67 benchmark. We tested our descriptor MC and the variants described in Sec. 3.5 on the task of recognizing the indoor-scenes present in the pictures of this dataset. Specifically: for a first set of experiments, we tested the descriptor MC-2048dims created by selecting from MC the 2048 most active features according to the criterion of Recursive Feature Elimination [45] (RFE). The feature selection was performed using ILSVRC 2010, by removing at each iteration 50% of the features. This lower-dimensional descriptor reduced the computational requirements and allowed us to easily perform an extensive set of evaluations. Table 4 summarizes the results of our experiments, and includes the recognition rates of several other methods presented in the literature. We can see that the plain descriptor MC-2048dims is already very competitive, yielding accuracy close to other state-of-the-art methods. All the local encoding variants of Sec. 3.5 are able to boost the accuracy. In particular we can notice that the method MC-2048dims+SPLPOOL L0L1L2 is superior to MC-2048dims+SPCAT L0L1 while producing a smaller descriptor; MC-2048dims+OBJPOOL is the best method as it produces a descriptor that is only twice as big as the original one and it yields the best recognition accuracy (+2.5% over MC-2048dims). Finally the full-dimensional MC +OBJPOOL yields an accuracy of 56% that, to the best of our knowledge, is the best published result for this benchmark.

Method	Accuracy
MC-2048dims	44.6
MC-2048dims + SPCAT L0L1	46.9
MC-2048dims + SPLPOOL L0L1L2	49.6
MC-2048dims + OBJPOOL	49.6
MC + OBJPOOL	55.9
Elfiky et al. 2012 [44]	48.9
Li et al. 2010 [17] (OBJECTBANK)	37.6

TABLE 4
Scene recognition on MIT 2007 using our descriptors and other methods in the literature. For our image representations, the classification model is a linear SVM. Our descriptor MC +OBJPOOL outperforms all prior methods on this test.

Method	Accuracy
CLASSEMES-BIT	17.6
PICODES	27.1
MC-BIT	34.8
MC	36.8
Xiao et al. 2010 [33]	38.00

TABLE 5
Scene recognition on SUN 397 using our descriptors and other methods in the literature. The classification model for our image representations is a linear SVM.

4.6.2 SUN 397

To conclude, we present experiments on the large-scale scene recognition benchmark SUN 397. We tested our binary descriptors CLASSEMES-BIT, PICODES and MC-BIT, and the real-valued MC. Table 5 shows our results. As already noticed in our prior evaluations, PICODES outperforms CLASSEMES-BIT, and the larger MC-BIT is the best performing one. The accuracy obtained with MC approaches the results provided by the method introduced in [33] which is a multiple-kernel combiner with 15 types of features, and thus orders of magnitudes more expensive to train and test and requiring much higher storage size.

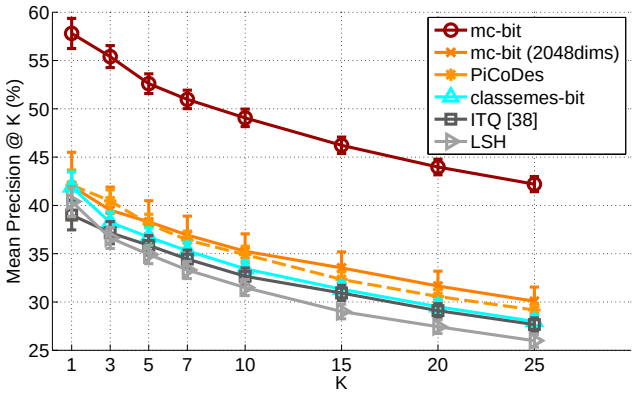

Fig. 4. Object-class search on ILSVRC2010: precision in retrieving images of a novel class from a dataset of 150,000 photos. For each query, the true positives are only 0.1% of the database. The classification model is a linear SVM.

4.7 Experiments on Object Search

4.7.1 ILSVRC 2010

We present here results for our motivating problem: fast novel-class recognition in a large-scale database. For this experiment we use again the ILSVRC2010 data set. However, this time for each class we learn a binary linear SVM using as positive examples all the images of that class in the ILSVRC2010 training set; as negative examples we use 4995 images obtained by sampling 5 images from each of the other 999 categories. Then we use the classifier to rank the 150,000 images of the test set. We measure performance in terms of mean precision at K, i.e., the average proportion of images of the relevant class in the top-K. Note that for each class, the database contains only 150 positive examples and 149,850 distractors from the other classes. Figure 4 shows the accuracy obtained with MC-BIT, which on this task outperforms by 15% CLASSEMES-BIT. We also compared our descriptors against 2048-bit codes learned by ITQ [38] and LSH (Random Projections) (we ran these methods on a compressed version of the representation PSI obtained via PCA). It can be seen that all our methods outperform these baselines, even for the same target dimensionality.

While the systems described in [2], [9] achieve higher multiclass recognition accuracy than our method on ILSVRC2010 where the classes are predefined (see Sec. 4.5.2), we point out that these approaches are not scalable in the context of real-time object-class search in large databases. Table 2 (column three) reports the storage required by different methods for a database containing 10M images. In their proposed form, [2] and [9] require to store high-dimensional real-valued vectors. Even if splitting the data across different machines, these approaches remain clearly not scalable in real scenarios. In [2], product quantization (PQ) is used to compress down the size of the data but even in this case a 10M-images database would require 610 GB. Our approach saves an order of magnitude of storage, resulting in the most scalable method in terms of memory utilization.

In addition, our system is also outperforming the other competing methods in terms of recognition time. In table 2 (column four) we show the average search time per image for a single object-class query, and indicates that our methods are by far the fastest. The system proposed by [2], which was relatively scalable in terms of storage, is the slowest one. Our approach provides a 10-fold or greater speedup over these systems and is the only one computing results in times acceptable for interactive search. We also note that sparse retrieval models and top-k ranking methods [46] could be used with our binary code to achieve further speedups on the problem of class-search.

5 Conclusions

In this paper we have presented three image descriptors, which measure the closeness of a given image to a set of high-level visual concepts, called basis classes. The concepts are either chosen a priori or automatically learned. We implement this similarity measure using the outputs of non-linear classifiers trained on an offline labeled dataset. We also propose methods to aggregate the locally-dependent outputs of the basis classifiers into a single feature vector, thus rendering the descriptor more robust to changes in size and position of the object of interest. Our descriptors are designed to be very compact, yet they are able to achieve state-of-the art accuracy even with simple linear classifiers. We tested and compared our descriptors on several challenging benchmarks, on several tasks: object categorization, scene recognition and novel object-class search. Thank to the compactness and the rich visual information incorporated into the descriptors, the proposed framework enables real-time object-class training and search in databases containing millions of images with good accuracy. The software for the extraction of all our descriptors is publicly available.

Acknowledgment

We are grateful to Andrew Fitzgibbon for his contribution in the design of classemes and PICODES. We thank Martin Szummer for discussions and Chen Fang for programming help. This research was funded in part by Microsoft Research and NSF CAREER award IIS-0952943.

References

[1] 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.

[2] J. Sánchez and F. Perronnin, "High-dimensional signature compression for large-scale image classification," in *CVPR*, 2011, pp. 1665–1672.
[3] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," Caltech, Tech. Rep. 7694, 2007.
[4] A. Berg, J. Deng, and L. Fei-Fei, "Large scale visual recognition challenge," 2010, http://www.image-net.org/challenges/LSVRC/2010/.
[5] P. Gehler and S. Nowozin, "On feature combination for multiclass object classification," in *ICCV*, 2009.
[6] N. Kumar, A. Berg, P. Belhumeur, and S. Nayar, "Attribute and simile classifiers for face verification," in *ICCV*, 2009.
[7] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth, "Describing objects by their attributes," in *CVPR*, 2009.
[8] C. H. Lampert, H. Nickisch, and S. Harmeling, "Learning to detect unseen object classes by between-class attribute transfer," in *CVPR*, 2009.
[9] Y. Lin, F. Lv, S. Zhu, M. Yang, T. Cour, K. Yu, L. Cao, and T. S. Huang, "Large-scale image classification: Fast feature extraction and svm training," in *CVPR*, 2011.
[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1106–1114.
[11] S. Maji and A. C. Berg, "Max-margin additive classifiers for detection," in *ICCV*, 2009.
[12] A. Vedaldi and A. Zisserman, "Efficient additive kernels via explicit feature maps," *PAMI*, 2011.
[13] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid, "Aggregating local image descriptors into compact codes," *IEEE Trans. on PAMI*, 2011.
[14] G. Wang, D. Hoiem, and D. Forsyth, "Learning image similarity from flickr using stochastic intersection kernel machines," in *ICCV*, 2009.
[15] L. Torresani, M. Szummer, and A. Fitzgibbon, "Efficient object category recognition using classemes," in *ECCV*, 2010.
[16] J. Deng, A. Berg, and F.-F. Li, "Hierarchical semantic indexing for large scale image retrieval," in *CVPR*, 2011.
[17] 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.
[18] A. Bergamo, L. Torresani, and A. Fitzgibbon, "Picodes: Learning a compact code for novel-category recognition," in *NIPS*, 2011, pp. 2088–2096.
[19] A. Bergamo and L. Torresani, "Meta-class features for large-scale object categorization on a budget," *CVPR*, vol. 0, pp. 3085–3092, 2012.
[20] D. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, vol. 60, no. 2, pp. 91–110, 2004.
[21] 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*, vol. 155, 2006.
[22] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *JMLR*, vol. 9, 2008.
[23] S. Bengio, J. Weston, and D. Grangier, "Label embedding trees for large multi-class tasks," in *NIPS*, 2010.
[24] T. Gao and D. Koller, "Discriminative learning of relaxed hierarchy for large-scale visual recognition," in *ICCV*, 2011.
[25] J. Deng, S. Satheesh, A. C. Berg, and F.-F. Li, "Fast and balanced: Efficient label tree learning for large scale object recognition," in *NIPS*, 2011, pp. 567–575.
[26] U. von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, Dec. 2007.
[27] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *NIPS*, 2001.
[28] J. C. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," in *Advances in Large Margin Classifiers*. MIT Press, 1999.
[29] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *CVPR*, 2006.
[30] B. Alexe, T. Deselaers, and V. Ferrari, "Measuring the objectness of image windows," *IEEE PAMI*, vol. 34, no. 11, pp. 2189–2202, 2012.
[31] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results," http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.
[32] A. Quattoni and A. Torralba, "Recognizing indoor scenes," in *CVPR*, 2009, pp. 413–420.
[33] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, "Sun database: Large-scale scene recognition from abbey to zoo," in *CVPR*, 2010, pp. 3485–3492.
[34] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, 2005.
[35] E. Shechtman and M. Irani, "Matching local self-similarities across images and videos," in *CVPR*, 2007.
[36] A. Vedaldi and B. Fulkerson, "VLFeat: An open and portable library of computer vision algorithms," http://www.vlfeat.org/, 2008.
[37] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," California Institute of Technology, Tech. Rep. 7694, 2007. [Online]. Available: http://authors.library.caltech.edu/7694
[38] Y. Gong and S. Lazebnik, "Iterative quantization: A procrustean approach to learning binary codes," in *CVPR*, 2011, pp. 817–824.
[39] "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.
[40] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," California Institute of Technology, Tech. Rep. 7694, 2007.
[41] J. Deng, A. C. Berg, K. Li, and F.-F. Li, "What does classifying more than 10, 000 image categories tell us?" in *ECCV*, 2010.
[42] H. Harzallah, F. Jurie, and C. Schmid, "Combining efficient object localization and image classification," in *ICCV*, 2009, pp. 237–244.
[43] Z. Song, Q. Chen, Z. Huang, Y. Hua, and S. Yan, "Contextualizing object detection and classification," in *CVPR*, 2011, pp. 1585–1592.
[44] N. M. Elfiky, J. Gonzàlez, and F. X. Roca, "Compact and adaptive spatial pyramids for scene recognition," *Image Vision Comput.*, vol. 30, no. 8, pp. 492–500, 2012.
[45] O. Chapelle and S. S. Keerthi, "Multi-class feature selection with support vector machines," *Proc. Am. Stat. Ass.*, 2008.
[46] M. Rastegari, C. Fang, and L. Torresani, "Scalable object-class retrieval with approximate and top-k ranking," in *ICCV*, 2011.

Alessandro Bergamo is a Ph.D. student in the Computer Science Department at Dartmouth College. He received his B.S. from the University of Padua and his M.S. from the University of Milan in Italy. His research interests spread between the areas of Machine Learning and Computer Vision, with special focus on the design of image descriptors and methods for efficient novel-class recognition and search in large-scale databases.

Lorenzo Torresani is an Assistant Professor in the Computer Science Department at Dartmouth College. He received a Laurea Degree in Computer Science with summa cum laude honors from the University of Milan in 1996, and an M.S. and a Ph.D. in Computer Science from Stanford University in 2001 and 2005, respectively. He has worked at several industrial research labs including Microsoft Research, Like.com and Digital Persona. His research is in computer vision and machine learning. In 2001, Torresani and his coauthors received the Best Student Paper Award at the IEEE Conference On Computer Vision and Pattern Recognition (CVPR). He is the recipient of a National Science Foundation CAREER Award.