Meta-Class Features for Large-Scale Object Categorization on a Budget

Alessandro Bergamo Lorenzo Torresani
Dartmouth College
Hanover, NH, U.S.A.
{aleb, lorenzo}@cs.dartmouth.edu

Abstract

In this paper we introduce a novel image descriptor enabling accurate object categorization even with linear models. Akin to the popular attribute descriptors, our feature vector comprises the outputs of a set of classifiers evaluated on the image. However, unlike traditional attributes which represent hand-selected object classes and predefined visual properties, our features are learned automatically and correspond to "abstract" categories, which we name meta-classes*. Each meta-class is a super-category obtained by grouping a set of object classes such that, collectively, they are easy to distinguish from other sets of categories. By using "learnability" of the meta-classes as criterion for feature generation, we obtain a set of attributes that encode general visual properties shared by multiple object classes and that are effective in describing and recognizing even novel categories, i.e., classes not present in the training set. We demonstrate that simple linear SVMs trained on our meta-class descriptor significantly outperform the best known classifier on the Caltech256 benchmark. We also present results on the 2010 ImageNet Challenge database where our system produces results approaching those of the best systems, but at a much lower computational cost.*

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 [9, 27]. 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. This is clearly reflected in the benchmarks of this field [13, 3], 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 of the recognition.

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 search and recognition of arbitrary categories in gigantic image collections, where the classes are not known at the time of the creation of the database. We propose to achieve this goal by means of a novel image descriptor enabling good recognition accuracy even 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. Furthermore, we show that our 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.

Finally, while multiclass recognition of a fixed set of categories is *not* our main motivating application, nevertheless we show that our approach achieves 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 [12] trained on the same low-level features used to learn our descriptor. On the 2010 ImageNet Challenge (ILSVRC2010) database, linear classification with our meta-class features achieves recognition accuracy only 10.3% lower than the winner of the competition [20], which is a system that was trained for a week using a powerful cluster of machines, a specialized hardware architecture for memory sharing, and a file system capable of handling terabytes of data; instead, our approach allows use to fit the entire ILSVRC2010 training and testing set in the RAM of a standard computer and produce results within a day using a budget PC.

In our approach we 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 effectively implements a nonlinear function of the original low-level features. As demonstrated in recent literature on object categorization [30, 12], these nonlinearities are critical to achieve good categorization accuracy with low-level features. 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. We are not the first to propose the idea of using the scores of nonlinear classifiers as features to achieve good recognition accuracy at low cost [29, 19]. However, the fundamental difference with our work is that in these prior systems the individual classifiers defining the descriptor are trained to recognize a hand-selected set of classes or visual properties. Our contribution is to replace these subjectively-chosen classes with learned "abstract" categories, i.e., categories that do not necessarily exist in the real-world but that capture salient common visual properties shared among many object classes. We refer to these abstract categories 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 set of object classes in the training set. Specifically, we hierarchically partition the set of training object classes into subsets 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. We demonstrate that our meta-class features greatly outperform classifier-based descriptors defined in terms of hand-selected classes [29], precisely because our abstract categories encode properties shared by many object classes and thus can produce better generalization on novel categories. Furthermore, we present for the first time categorization results using classifier-based descriptors on the large-scale ILSVRC2010 database and study their efficiency advantages compared to prior work.

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 [12], 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 [9]. 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 [22, 31]: for many popular kernels in computer vision, these methods provide analytical mappings to slightly 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 [27]: 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 [27, 15]. An exception is the work of Lin et al. [20] where the high storage and I/O costs caused by their high-dimensional descriptor were absorbed by a large system infrastructure consisting of Apache Hadoop to distribute computation and storage over many machines.

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 [33, 29, 7] or as the response map to a set of detectors [19]. 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 [16, 11, 17], which are semantic characteristics selected by humans as associated to the classes to recognize.

Our approach is most closely related to this third line of work, as we also represent images in terms of the outputs of classifiers learned for a set of basis classes. However, we demonstrate that better accuracy can be achieved by learning the basis classes as abstract visual categories useful for recognition rather than by hand-selecting them. Our meta-classes are similar in spirit to PiCoDes [4], which are binary descriptors also encoding a set of learned, abstract attributes. However, the abstract classifiers of PiCoDes are jointly computed via an expensive iterative optimization, which in practice can be run only for very compact dimensionalities (the largest PiCoDes contain 2048 bits and required several weeks to be learned). Instead, our meta-classes are efficiently defined via recursive application of spectral clustering and the classifiers recognizing these meta-classes can be trained in parallel. Thus, our approach can easily scale to much larger descriptor sizes.

3. Technical Approach

In this section we provide a detailed description of the procedure used to train our image descriptor. Our representation is learned from a labeled dataset of images $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, where $\mathbf{x}_i$ denotes the i-th image in the database and y_i indicates the class label of the object present in the photo.

3.1. The meta-class feature

We start by introducing the model of our meta-class feature. We indicate with $\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x}), \ldots, h_C(\mathbf{x})]^T$ our C-dimensional meta-class descriptor extracted from an image $\mathbf{x}$. Each descriptor entry $h_c(\mathbf{x})$ is the output of a binary classifier evaluated on $\mathbf{x}$. The hypothesis h_c is learned from a training set $\mathcal{D}_c$ containing images of meta-class c as positive examples, and of other meta-classes as negative images. In subsection 3.2, we discuss how the training set $\mathcal{D}_c$ is obtained from the original set $\mathcal{D}$ for each meta-class c.

As classification model for h_c, we use a variant of the LP-β classifier [12], which has been shown to yield state-of-the-art results on several categorization benchmarks. The LP-β model is computed as a linear combination of M nonlinear classifiers, each trained on a different low-level feature vector $\mathbf{f}_m(\mathbf{x})$ for $m = 1, \ldots, M$. In our implementation we use $M = 13$ low-level features, which are described in the experimental section of the paper. Note that while extraction of the low-level features $\mathbf{f}_m(\mathbf{x})$ is needed to calculate the descriptor $\mathbf{h}(\mathbf{x})$ for an image, storage of these low-level features is not necessary for the subsequent recognition, which is done exclusively using the low-dimensional vector $\mathbf{h}(\mathbf{x})$. While the original LP-β model of Gehler and Nowozin employs kernels to render the classifier nonlinear, in this work we approximate the kernel distances by adopting the "lifting" method of Vedaldi and Zisserman [31], which allows us to obtain a more efficient scheme for training the meta-class classifiers with very little loss in accuracy: the idea of this approach is to lift each feature vector $\mathbf{f}_m(\mathbf{x}) \in \mathbb{R}^{d_m}$ to a slightly higher-dimensional feature space via an explicit map $\Psi_m : \mathbb{R}^{d_m} \longrightarrow \mathbb{R}^{d_m(2r+1)}$ (where r is a small positive integer) such that inner products in this space approximate well the nonlinear kernel distance $K_m()$, i.e., $\langle \Psi_m(\mathbf{f}_m(\mathbf{x})), \Psi_m(\mathbf{f}_m(\mathbf{x}'))\rangle \approx K_m(\mathbf{f}_m(\mathbf{x}), \mathbf{f}_m(\mathbf{x}'))$. For the family of additive kernels the map can be analytically computed and good approximations can be obtained even when setting r to small values (here we use $r = 1$). This trick allows us to approximate the traditional LP-β classifier, which is computationally very expensive to train, with a linear combination of *linear* classifiers, which can be learned much more efficiently. Each meta-class classifier $h_c(\mathbf{x})$ is an instance of this efficient model:

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

Following the customary training scheme of LP-β, we first learn the parameters $\{\mathbf{w}_{m,c}, b_{m,c}\}$ for each feature m independently by training the hypothesis $h_{m,c}(\mathbf{x}) = \left[\mathbf{w}_{m,c}^T \Psi_m(\mathbf{f}_m(\mathbf{x})) + b_{m,c}\right]$ using the traditional SVM large-margin objective on training set $\mathcal{D}_c$. In a second step, we optimize over parameter vector $\boldsymbol{\beta}_c = [\beta_{1,c}, \ldots, \beta_{m,c}]^T$ constraining $\sum_m \beta_{m,c} = 1$ (see [12] for further details).

We found that approximating the kernel distance via explicit maps decreases the overall accuracy of the classifier by only a few percentage points, but it allows us to speed up the training procedure by several orders of magnitude.

3.2. Learning the meta-classes

In this section we describe the procedure to learn the meta-classes. Our method is an instance of the algorithm for label tree learning described in [2]. 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. We adopt the label tree training procedure from this prior work, but use it to learn meta-classes, i.e, set 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 $\mathcal{D}$. 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 $h_{(\ell^L,\ell^R)}(\mathbf{x})$ trained to distinguish these two meta-classes makes as few mistakes as possible. Unfortunately we do to know the accuracy of the classifier before training it. In principle we could train a classifier for each of the possible $(|\ell|(|\ell|-1)/2-1)$ non-trivial partitions of ℓ and then measure accuracy on a separate validation set. However, this would be prohibitively expensive. Instead, we can 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 $\mathcal{D}$. 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 $\mathcal{D}^{\text{val}}$: A_{ij} gives the number of samples of class i in $\mathcal{D}^{\text{val}}$ that have been predicted to belong to class j[1]. 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:

$$E(\ell^L) = \sum_{i,j \in \ell^L} B_{ij} + \sum_{p,q \in \ell - \ell^L} B_{pq} \,. \qquad (2)$$

The objective encourages to include within the same label 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. 2 can be formulated as a graph partitioning problem [32]. We compute the solution ℓ^L by applying spectral clustering [23] to the matrix B: this is equivalent to solving a relaxed, normalized version of eq. 2 that penalizes unbalanced partitions. Once the partition is computed, we train a classifier of the model described in section 3.1 on the resulting binary classification problem. In other words, let I^ℓ denote the indices of training examples having class labels in ℓ. Then, we form the labeled set $\mathcal{D}_{(\ell^L,\ell^R)} = \{(\mathbf{x}_i, +1) : i \in I^{\ell^L}\} \cup \{(\mathbf{x}_i, -1) : i \in I^{\ell^R}\}$ and use it to train meta-class classifier $h_{(\ell^L,\ell^R)}(\mathbf{x})$. We repeat this process recursively on each node until it contains a single class label, i.e., $|\ell| = 1$.

The final meta-class descriptor is defined as the concatenation of all meta-class classifiers learned in the creation of the tree. Optionally, the one-vs-the-rest classifiers $\hat{h}_1, \ldots, \hat{h}_{|\ell_{\mathcal{D}}|}$ can also be included as part of the descriptor: unlike the meta-class classifiers, these are trained to recognize real object classes and are of the same form as the "classeme" classifiers described in [29]. In our experiments we demonstrate that meta-class features produce better generalization performance than classemes on novel categories, probably due to their ability to capture properties shared by many classes rather than attributes of individual categories.

4. Experiments

In this section we experimentally compare our approach to several competing methods on the Caltech256 [13] and ILSVRC2010 [3] benchmarks.

4.1. Implementation of meta-class feature learning

We begin by describing the setup used to learn our descriptor. Our meta-class classifiers are based on the same low-level features used in [29]: color GIST [24], oriented and unoriented HOG [6], SSIM [28] and SIFT [21]. For each feature, we compute a spatial pyramid histogram [18]. This yields a total of $M = 13$ pyramid levels, each treated as a separate feature vector (for details, see [29]).

We "lift-up" each feature vector using the explicit feature map of Vedaldi and Zisserman [31] to approximate the histogram intersection kernel, with parameter $r = 1$. Thus, we can think of each nonlinear meta-class classifier as a linear hypothesis learned in a space of dimensionality $D = d(2r+1)$, where $d = \sum_{m=1}^{M} d_m$ is the dimensionality of the features concatenated. In our experiments , $d = 17360$, and therefore $D = 52080$.

As training set for the descriptor learning we use a subset of ImageNet [9] consisting of 8000 randomly sampled synsets (which from now we also refer to as "classes"). 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 perform 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. The number of examples for each training synset varies from a minimum of 40 to a maximum of 1000. We use an independent validation set of 80 examples per class to learn the confusion matrix A. As described in the previous section, we recursively apply spectral clustering to the confusion matrix creating a label tree consisting of 7458 internal nodes. Then, for each node with label set

[1] We assume the traditional winner-take-all strategy for multiclass classification, where each example is assigned to the class with the highest classifier score

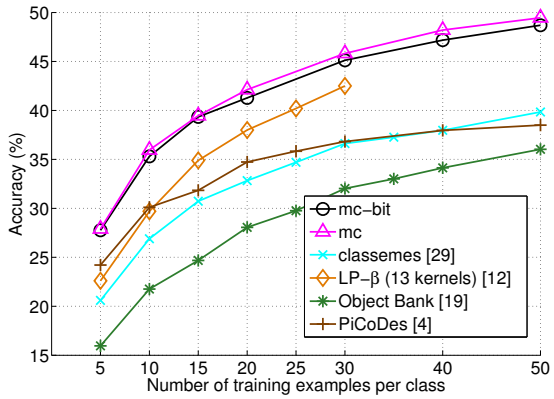

Figure 1. Multiclass object categorization accuracy on Caltech256. A simple linear SVM applied to our **mc** or **mc-bit** descriptor outperforms the state-of-the-art LP-β classifier and is also orders of magnitude faster to both train and test.

ℓ we learn a meta-classifier $h_{(\ell^L,\ell^R)}$ that discriminates the meta-classes of its children. In order to reduce the computational cost of the learning, we cap the number of training examples for each meta-class at 2000. We also include in the descriptor the outputs of the 8000 one-vs-the-rest classifiers $\hat{h}_c$ (the classemes), thus producing a feature vector containing a total of 15,458 entries for each image.

We convert the raw score of each h_c into a probabilistic output by means of Platt's scaling [25], by replacing $h_c(\boldsymbol{x})$ with $\sigma_c(h_c(\boldsymbol{x}))$ in the descriptor. We learn the two parameters of the sigmoid function for each meta-class using the validation set. As already noted in [7], we found that this sigmoidal normalization yields a considerable boost in accuracy when using the outputs of classifiers as descriptors for novel category recognition, probably as it makes the range of classification scores more homogeneous and reduces outlier values. We refer to this version of the meta-class descriptor as **mc**. We have also tried to aggressively quantize the descriptor by thresholding the raw outputs of the meta-class classifiers at 0, i.e., using the information bit $[h_c(\boldsymbol{x}) > 0]$ as feature. We denote this binary descriptor as **mc-bit**.

4.2. Multiclass recognition on Caltech256

We now present experiments obtained with our descriptor on the Caltech256 benchmark. We adopt a simple one-vs-the-rest linear SVM as multi-class classifier, performing prediction using the winner-take-all strategy. We use 5-fold cross validation for model selection. We use a test set of 25 examples per class and, as customary, we calculate the accuracy as the mean of the diagonal of the confusion matrix. We compare our descriptor to the following approaches:

- the original classeme vector [29] which is defined by 2659 object classifiers trained on weakly-labeled images obtained from an image search engine;
- PiCoDes [4], which contain the binary outputs of 2048 classifiers jointly optimized for linear classification accuracy using a portion of the ImageNet dataset as training set;
- the object bank descriptor [19], which includes spatial pyramid histograms of the responses of a large set of object detectors applied to the image. The dimensionality of this vector is 44604, much larger than that of our descriptor;
- the state-of-the-art multiple kernel combiner LP-β [12] trained with *exact* intersection kernel distances using our set of 13 low-level features.

Figure 1 summarizes the results as a function of the number of training examples per class. We can see that our descriptors greatly outperforms all the other representations. The binarized version is only 1% worse than the real-valued **mc** descriptor while being 32 times more compact (using 1 bit for each entry of the vector the storage size for **mc-bit** is less than 2KB per image). In particular, we would like to stress that our descriptor yields accuracy improvements of up to +6% in absolute value over LP-β , which to the best of our knowledge represents the best performing classifier on Caltech256. Moreover this improvement is obtained by using a simple linear model, which is very efficient to train and – even more relevant for our motivating application – it is more than two orders of magnitude faster at test-time compared to LP-β as it involves only a dot-product instead of expensive kernel distance computations. Note that while one could adopt the explicit map trick to train and test more efficiently the LP-β classifier on the Caltech256 classes, this would not solve the large storage problem: the low-level features used by this approach would require 100 times more memory space than **mc-bit**. This prevents the use of such an approach in large scale datasets, such as the one considered in the next subsection.

4.3. Large-scale categorization on ILSVRC2010

We now move on to the problem of multiclass recognition on the ILSVRC2010 data set. This database includes a subset of 1000 synsets. Again, we remind the reader that these classes are disjoint from the training categories used to learn the descriptor. The training set contains a variable number of examples per class, from a minimum of 619 to a maximum of 3047 yielding in total about 1.2M examples. The test set includes 150,000 images, with 150 examples per category, and the validation set consists of 50 images for each class. Such massive amount of data poses new challenges and issues that are not present in smaller databases, as already noted in [8, 27, 20]. Yet, our binary **mc-bit** features render the learning on this database relatively fast to accomplish even on a budget PC for the following reasons. First, since we need only one bit per dimension we can store the entire ILSVRC2010 training set in only $(15458 \times 1.2\text{M})/8$ Bytes = 2.16 GigaBytes. This low memory requirement allows us to use efficient software

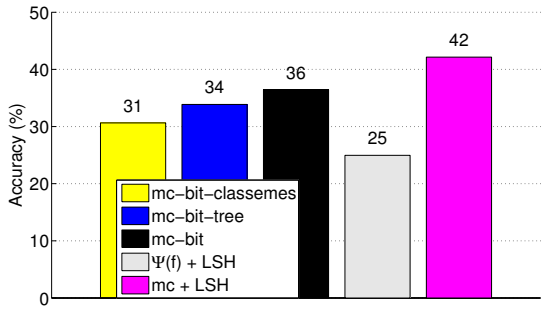

Figure 2. Top-1 multiclass recognition accuracy on ILSVRC2010 for different descriptors. From left to right: **mc-bit-classemes** is the part of our binarized descriptor that contains the features generated by the one-vs-the-rest classemes classifiers; **mc-bit-tree** is the part of our binarized descriptor that contains the features produced by the meta-class classifiers of the label tree; **mc-bit**: our full binarized descriptor; **LSH on lifted-up features**: LSH applied to $\Psi(\mathbf{f}(\boldsymbol{x}))$; **LSH on mc**: LSH applied to our real-valued descriptor.

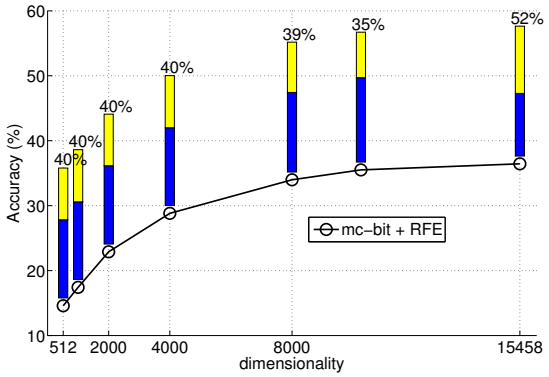

Figure 3. Multiclass recognition accuracy as a function of **mc-bit** dimensionality on ILSVRC2010. We use Recursive Feature Elimination to reduce the dimensionality of our **mc-bit** descriptor. The percentage at each dimensionality indicates the proportion of classeme features retained in the descriptor. Although initially the full descriptor contains more classemes than meta-classes, the majority of features selected at each step are meta-classes.

for batch training of linear SVMs. In particular we had success with LIBLINEAR [10], simply modifying the code to support input data in bitmap format. Moreover with binary data the dot-products between examples no longer require floating-point calculations but only a logical AND and a sparse summation. Again we use the one-vs-the-rest strategy to perform multiclass classification. To speed-up the learning we reduce the negative training set by sampling $n = 150$ examples per class. According to our study, this sampling does not cause a significant drop in performance.

In figure 2 we analyze the performance obtained with our descriptor on the ILSVRC2010 test set. We also provide results achieved with the individual subcomponents of **mc-bit**: "**mc-bit-tree**" which consists of the 7458 meta-class classifiers learned for the inner nodes of the label tree, versus "**mc-bit-classemes**", which contains the outputs of the 8000 one-vs-the-rest classeme classifiers. 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 (36.44%), suggesting that there is value in using both subcomponents.

We also report a baseline corresponding to a binary descriptor obtained by applying LSH [14] to the lifted-up low-level features: we first apply PCA to $\Psi(\mathbf{f}(\boldsymbol{x})) \in \mathbb{R}^{52080}$ compressing down the data to 9000 dimensions, and then take 15458 random LSH projections to produce a binary vector of the same length as our descriptor. As we can see, for the same descriptor size, the resulting performance is much lower than that produced with our method.

As we saw before in figure 1, the real-valued version of our descriptor tends to yield higher accuracy. To use the additional information contained in the real values of our descriptor while still exploiting the benefit of binary features, we apply LSH to **mc** and train on the resulting binary descriptor. Using $200,000$ random LSH projections the training set size for each class becomes 3.6 GigaBytes, when subsampling the negative examples as discussed above. Using this approach, our system achieves an accuracy of 42.15%. The performance of our method according to the top-5 accuracy measure of ILSVRC2010 is 64.01%, which is only 10.29% worse than the best published result. Note that our system would have ranked 3rd in the ILSVRC2010 challenge, just behind the systems of [27] and [20], which are orders of magnitude more costly.

We would like to point out that the training and testing of our system in this experiment can be done on a standard computer thanks to the low memory and training-time requirements. Training a single one-vs-the-rest using our binary meta-class 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 [20] required a week of training with a powerful cluster of machines and specialized hardware.

To further study the relative importance of the meta-class versus classeme features in our combined **mc-bit** descriptor, we performed a feature selection experiment: starting from the full binary descriptor size, we reduce the dimensionality by means of Recursive Feature Elimination [5] (RFE) where at each iteration we remove 50% of the features. Figure 3 shows the resulting accuracy on the test set as a func-

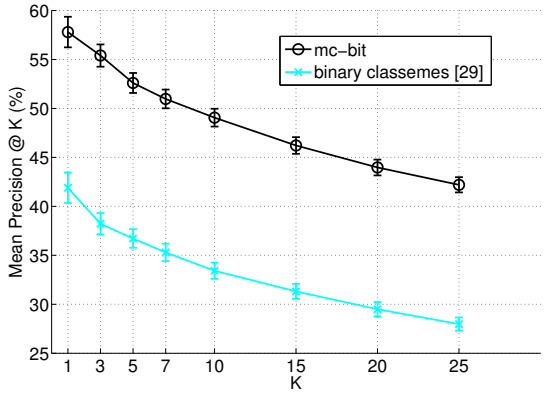

Figure 4. Object-class search on ILSVRC2010: accuracy in retrieving images of a novel class from a dataset of 150,000 photos. For each query class, the true positives are only 0.1% of the database. Our descriptor significantly outperforms classemes, which in [29] were also tested on such task.

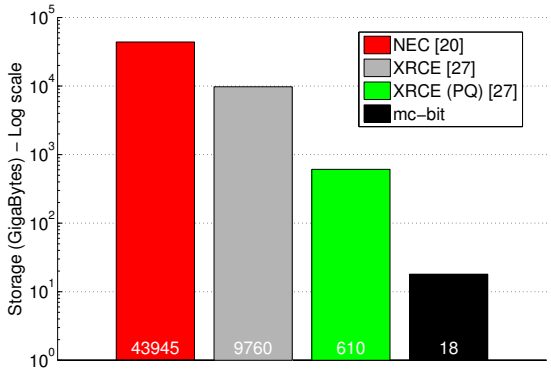

Figure 5. Storage requirements for 10M images with different feature representations (note the log scale). Our descriptors outperforms the top-method for ILSVRC2010 in terms of scalability to large databases.

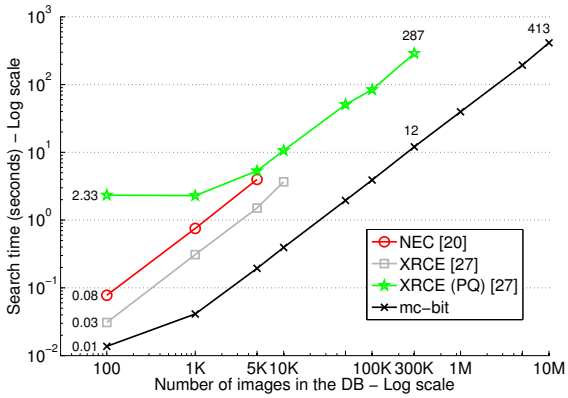

Figure 6. Object-class search time as a function of the number of images in the database, using a machine with 20 GB of memory. Our approach is significantly faster than the competing approaches and is the only one that allows large databases to be kept in the memory of standard computers.

tion of the dimensionality. For each feature elimination step we display the proportion of features in the descriptor that are one-vs-the-rest classemes. In agreement with the results previously shown in figure 2, this plot suggests that meta-class features are found to be more effective as the RFE procedure retains consistently more meta-class features than classemes at each selection step.

4.4. Object-class search on ILSVRC2010

Finally we present results for our motivating problem: fast novel-class recognition in a large-scale database. For this experiment we use again the ILSVRC2010 data set. 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 performance obtained with **mc-bit**, which on this task outperforms by 15% the binary classeme vector of [29].

While the systems described in [27, 20] achieve higher multiclass recognition accuracy than our method on ILSVRC2010 where the classes are predefined, we point out that they are not scalable in the context of object-class search in large databases. Figure 5 reports the storage required by different methods for a database containing 10M images. In their proposed form, [27] and [20] require to store high-dimensional real-valued vectors: the feature vector dimensionality is 1.18M for the former and 524K for the latter. Thus, a 10M data set would require 44TByte and 10 TByte, respectively. Even if splitting the data across different machines, these approaches remain clearly not scalable in real scenarios. In [27], product quantization is used to compress down the size of the data, at the expense of a drop in accuracy of about 1%. With this compression, a 10M 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 figure 6 we show the average search time for a single object-class query as a function of the number of images contained in the database. For this experiment we used a computer with a CPU Intel Xeon X5675@3.07GHz and 20 GB of memory. As shown in figure 5 only our representation allows the entire ImageNet database to be kept in the memory of this computer. The plot of figure 6 indicates that our method is by far the fastest. The system proposed by [27], 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 [26] could be used with our binary code to achieve further speedups on the problem of class-search.

5. Conclusions

We have introduced a novel image descriptor achieving excellent recognition accuracy even with simple linear classifiers. The features of the descriptor are learned from labeled data and represent general visual properties shared by many object classes. We demonstrate that they yield improved recognition accuracy over hand-selected attributes. Thanks to the compactness of this representation, it is possible to keep the entire ILSVRC2010 database in the memory of a budget PC, and train classifiers for all 1000 classes in 14 hours on a single core machine. The resulting accuracy is 10% from the the state-of-the-art. We know of no other method that can achieve this level of accuracy while offering such efficiency and scalability to large data sets.

Additional material including software to extract our descriptor from images may be obtained from [1].

6. Acknowledgements

We are grateful to Jia Deng and Florent Perronnin for answering questions about their systems. This research was funded in part by NSF CAREER award IIS-0952943.

References

[1] http://vlg.cs.dartmouth.edu/metaclass. 8

[2] S. Bengio, J. Weston, and D. Grangier. Label embedding trees for large multi-class tasks. In *NIPS*, 2010. 3

[3] A. Berg, J. Deng, and L. Fei-Fei. Large scale visual recognition challenge, 2010. http://www.image-net.org/challenges/LSVRC/2010/. 1, 4

[4] A. Bergamo, L. Torresani, and A. Fitzgibbon. Picodes: Learning a compact code for novel-category recognition. In *NIPS*, pages 2088–2096. 2011. 3, 5

[5] O. Chapelle and S. S. Keerthi. Multi-class feature selection with support vector machines. *Proc. Am. Stat. Ass.*, 2008. 6

[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 4

[7] J. Deng, A. Berg, and F.-F. Li. Hierarchical semantic indexing for large scale image retrieval. In *CVPR*, 2011. 3, 5

[8] 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. 5

[9] 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. 1, 2, 4

[10] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *JMLR*, 9, 2008. 6

[11] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing objects by their attributes. In *CVPR*, 2009. 3

[12] P. Gehler and S. Nowozin. On feature combination for multiclass object classification. In *ICCV*, 2009. 2, 3, 5

[13] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report 7694, Caltech, 2007. 1, 4

[14] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *STOC 1998*, New York, NY, USA, 1998. ACM Press. 6

[15] 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. 2

[16] N. Kumar, A. Berg, P. Belhumeur, and S. Nayar. Attribute and simile classifiers for face verification. In *ICCV*, 2009. 3

[17] C. H. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*, 2009. 3

[18] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006. 4

[19] 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. 2, 3, 5

[20] 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. 2, 5, 6, 7

[21] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004. 4

[22] S. Maji and A. C. Berg. Max-margin additive classifiers for detection. In *ICCV*, 2009. 2

[23] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, 2001. 4

[24] 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. 4

[25] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*. MIT Press, 1999. 5

[26] M. Rastegari, C. Fang, and L. Torresani. Scalable object-class retrieval with approximate and top-k ranking. In *ICCV*, Nov. 2011. 8

[27] J. Sánchez and F. Perronnin. High-dimensional signature compression for large-scale image classification. In *CVPR*, pages 1665–1672, 2011. 1, 2, 5, 6, 7

[28] E. Shechtman and M. Irani. Matching local self-similarities across images and videos. In *CVPR*, 2007. 4

[29] L. Torresani, M. Szummer, and A. Fitzgibbon. Efficient object category recognition using classemes. In *ECCV*, 2010. 2, 3, 4, 5, 7

[30] M. Varma and D. Ray. Learning the discriminative power-invariance trade-off. In *ICCV*, pages 1–8, 2007. 2

[31] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. *PAMI*, 2011. 2, 3, 4

[32] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, Dec. 2007. 4

[33] G. Wang, D. Hoiem, and D. Forsyth. Learning image similarity from flickr using stochastic intersection kernel machines. In *ICCV*, 2009. 3