Learning and Using Taxonomies For Fast Visual Categorization

Gregory Griffin and Pietro Perona
California Institute of Technology
Computation and Neural Systems Dept.
Pasadena, CA 91125, USA
{greg,perona}@vision.caltech.edu

Abstract

The computational complexity of current visual categorization algorithms scales linearly at best with the number of categories. The goal of classifying simultaneously $N_{\rm cat} = 10^4 - 10^5$ visual categories requires sub-linear classification costs. We explore algorithms for automatically building classification trees which have, in principle, $\log N_{\rm cat}$ complexity. We find that a greedy algorithm that recursively splits the set of categories into the two minimally confused subsets achieves 5-20 fold speedups at a small cost in classification performance. Our approach is independent of the specific classification algorithm used. A welcome by-product of our algorithm is a very reasonable taxonomy of the Caltech-256 dataset.

1. Introduction

Much progress has been made during the past 10 years in approaching the problem of visual recognition. The literature shows a quick growth in the scope of automatic classification experiments: from learning and recognizing one category at a time until year 2000 [4, 24] to a handful around year 2003 [26, 8, 14] to $\sim$ 100 in 2006 [11, 10, 5, 21, 22, 28, 11]. While some algorithms are remarkably fast [9, 24, 10] the cost of classification is still at best linear in the number of categories; in most cases it is in fact quadratic since one-vs-one discriminative classification is used in most approaches. There is one exception: cost is logarithmic in the number of models for Lowe [18]. However Lowe's algorithm was developed to recognize specific objects rather than categories. Its speed hinges on the observation that local features are highly distinctive, so that one may index image features directly into a database of models which is organized like a tree [2]. In the more general case of visual category recognition, local features are not very distinctive, hence one cannot take advantage of this insight.

Humans can recognize between 10^4 and 10^5 object categories [3] and this is a worthwhile and practical goal for machines as well. It is therefore important to understand how to scale classification costs sub-linearly with respect to the number of categories to be recognized. It is quite intuitive that this is possible: when we see a dog we are not for a moment considering the possibility that it might be classified as either a jet-liner or an ice cream cone. It is reasonable to assume that, once an appropriate hierarchical taxonomy is developed for the categories in our visual world, we may be able to recognize objects by descending the branches of this taxonomy and avoid considering irrelevant possibilities. Thus, tree-like algorithms appear to be a possibility worth considering, although formulations need to be found that are more 'holistic' than Beis and Lowe's

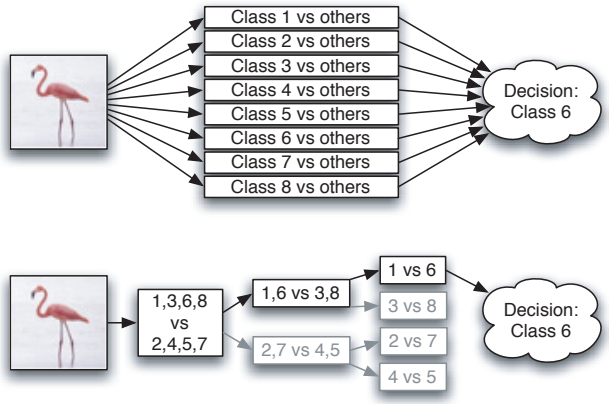

Figure 1. A typical one-vs-all multi-class classifier (top) exhaustively tests each image against every possible visual category requiring $N_{\rm cat}$ decisions per image. This method does not scale well to hundreds or thousands of categories. Our hierarchical approach uses the training data to construct a taxonomy of categories which corresponds to a tree of classifiers (bottom). In principle each image can now be classified with as few as $\log_2 N_{\rm cat}$ decisions. The above example illustrates this for an unlabeled test image and $N_{\rm cat} = 8$. The tree we actually employ has slightly more flexibility as shown in Fig. 4

978-1-4244-2243-2/08/$25.00 ©2008 IEEE

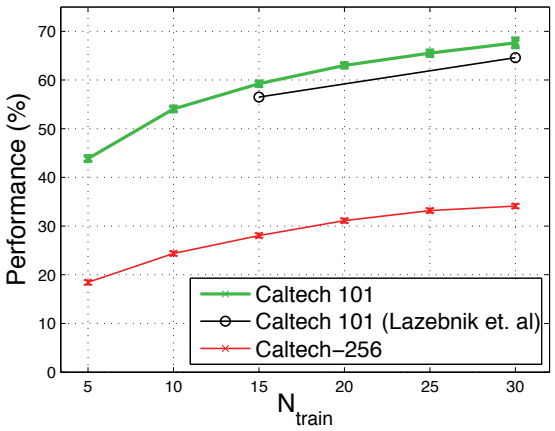

Figure 2. Performance comparison between Caltech-101 and Caltech-256 datasets using the Spatial Pyramid Matching algorithm of Lazebnik et. al [13]. The performance of our implementation is almost identical to that reported by the original authors; any performance difference may be attributed to a denser grid used to sample SIFT features. This illustrates a standard non-hierarchical approach where authors mainly present the number of training examples and the classification performance, without also plotting classification speed.

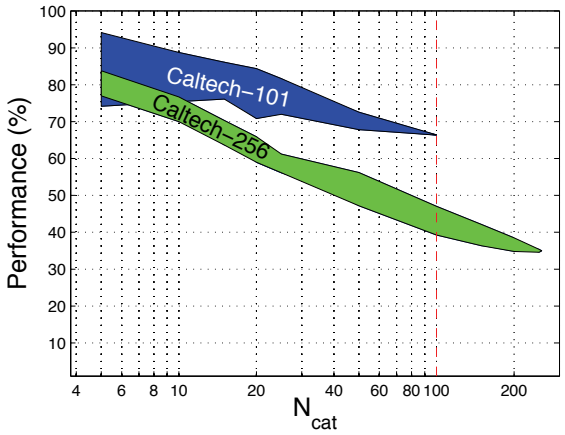

Figure 3. In general the Caltech-256 [12] images are more difficult to classify than the Caltech-101 images. Here we fix $N_{\text{train}} = 30$ and plot performance of the two datasets over a random mix of N_{cat} categories chosen from each dataset. The solid region represents a range of performance values for 10 randomized subsets. Even when the number of categories remains the same, the Caltech-256 performance is lower. For example at $N_{\text{cat}} = 100$ the performance is $\sim 60\%$ lower (dashed red line).

feature-based indexing [2].

Here we explore one such formulation. We start by considering the confusion matrix that arises in one-vs-all discriminative classification of object categories. We postulate that the structure of this matrix may reveal which categories are more strongly related. In Sec. 3 we flesh out this heuristic and to produce taxonomies. In Sec. 4 we propose a mechanism for automatically splitting large sets of categories into cleanly separated subsets, an operation which may be repeated obtaining a tree-like hierarchy of classifiers. We explore experimentally the implications of this strategy, both in terms of classification quality and in terms of computational cost. We conclude with a discussion in Sec. 5.

2. Experimental Setup

The goal of our experiment is to compare classification performance and computational costs when a given classification algorithm is used in the conventional one-vs-many configuration vs our proposed hierarchical cascade (see Fig. 1).

The choice of the image classifier is somewhat arbitrary for the purposes of this study. We decided to use the popular Spatial Pyramid Matching technique of Lazebnik et al. [13] because of its high performance and ease of implementation. We summarize our implementation in Sec.2.2. Our implementation performs as reported by the original authors on Caltech-101. As expected, typical performance on Caltech-256 [12] is lower than on Caltech-101 [15] (see Fig. 2). This is due to two factors: the larger number of categories and the more challenging nature of the pictures themselves. For example some of the Caltech-101 pictures are left-right aligned whereas the Caltech-256 pictures are not. On average a random subset of N_{cat} categories from the Caltech-256 is harder to classify than a random subset of the same number of categories from the Caltech-101 (see Fig. 3).

Other authors have achieved higher performance on the Caltech-256 than we report here, for example, by using a linear combination of multiple kernels [23]. Our goal here is not to achieve the best possible performance but to illustrate how a typical algorithm can be accelerated using a hierarchical set of classifiers.

2.1. Training and Testing Data

The Caltech-256 image set is used for testing and training. We remove the *clutter* category from Caltech-256 leaving a total of $N_{\text{cat}} = 256$ categories.

2.2. Spatial Pyramid Matching

First each image is desaturated, removing all color information. For each of these black-and-white images, SIFT features [18] are extracted along a uniform 72x72 grid using software that is publicly available [20]. An M-word feature vocabulary is formed by fitting a Gaussian mixture model to 10,000 features chosen at random from the training set.

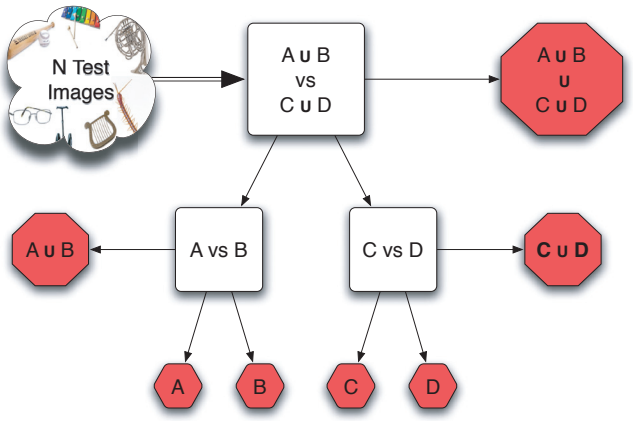

Figure 4. A simple hierarchical cascade of classifiers (limited to two levels and four categories for simplicity of illustration). We call A, B, C and D four sets of categories as illustrated in Fig 5. Each white square represents a binary *branch classifier*. Test images are fed into the top node of the tree where a classifier assigns them to either the set A ∪ B or the set C ∪ D (white square at the center-top). Depending on the classification, the image is further classified into either A or B, or C or D. Test images ultimately terminate in one of the 7 red octagonal nodes where a conventional multi-class *node classifier* makes the final decision. For a two-level $\ell = 2$ tree, images terminate in one of the 4 lower octagonal nodes. If $\ell = 0$ then all images terminate in the top octagonal node, which is equivalent to conventional non-hierarchical classification. The tree is not necessarily perfectly balanced: A, B, C and D may have different cardinality. Each branch or node classifier is trained exclusively on images extracted from the sets that the classifier is discriminating. See Sec. 4 for details.

This model maps each 128-dimensional SIFT feature vector to a scalar integer $m = 1..M$ where $M = 200$ is the total number of Gaussians. The choice of clustering algorithm does not seem to affect the results significantly, but the choice of M does. The original authors [13] find that 200 visual words are adequate.

At this stage every image has been reduced to a 72x72 matrix of visual words. This representation is reduced still further by histogramming over a coarse 4x4 spatial grid. The resulting 4x4xM histogram counts the number of times each word 1..M appears in each of the 16 spatial bins. Unlike a bag-of-words approach [11], coarse-grained position information is retained as the features are counted.

The matching kernel proposed by Lazebnik et al. finds the intersection between each pair of 4x4xM histograms by counting the number of common elements in any two bins. Matches in nearby bins are weighed more strongly than matches in far-away bins, resulting in a single match score for each word. The scores for each word are then summed to get the final overall score. We follow this same procedure resulting in a kernel K that satisfies Mercer's condition [11]

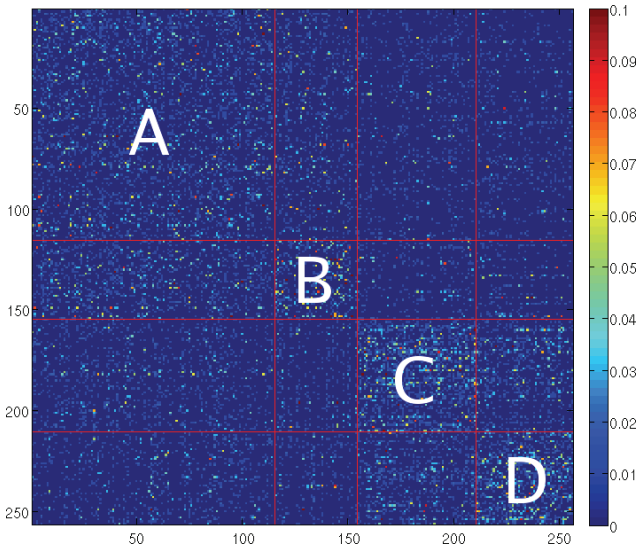

Figure 5. Top-down grouping as described in Sec. 3. Our underlying assumption is that categories that are easily confused should be grouped together in order to build the branch classifiers in Fig 4. First we estimate a confusion matrix using the training set and a leave-one-out procedure. Shown here is the confusion matrix for $N_{\text{train}} = 10$, with diagonal elements removed to make the off-diagonal terms easier to see.

and is suitable for training an SVM.

2.3. Measuring Performance

Classification performance is measured as a function of the number of training examples. First we select a random but disjoint set of N_{train} and N_{test} training and testing images from each class. All categories are sampled equally, ie. N_{train} and N_{test} do not vary from class to class.

Like Lazebnik et al. [13] we use a standard multi-class method consisting of a Support Vector Machine (SVM) trained on the Spatial Pyramid Matching kernel in a one-vs-all classification scheme. The training kernel has dimensions $N_{\text{cat}} \cdot N_{\text{train}}$ along each side. Once the classifier has been trained, each test image is assigned to exactly one visual category by selecting the one-vs-all classifier which maximizes the margin.

The confusion matrix $\mathcal{C}_{ij}$ counts the fraction of test examples from class i which were classified as belonging to class j. Correct classifications lie along the diagonal $\mathcal{C}_{ii}$ so that the cumulative performance is the mean of the diagonal elements. To reduce uncertaintly we average the matrix obtained over 10 experiments using different randomized training and testing sets. By inspecting the off-diagonal elements of the confusion matrix it is clear that some categories are more difficult to discriminate than other categories. Upon this observation we build a heuristic that creates an efficient hierarchy of classifiers.

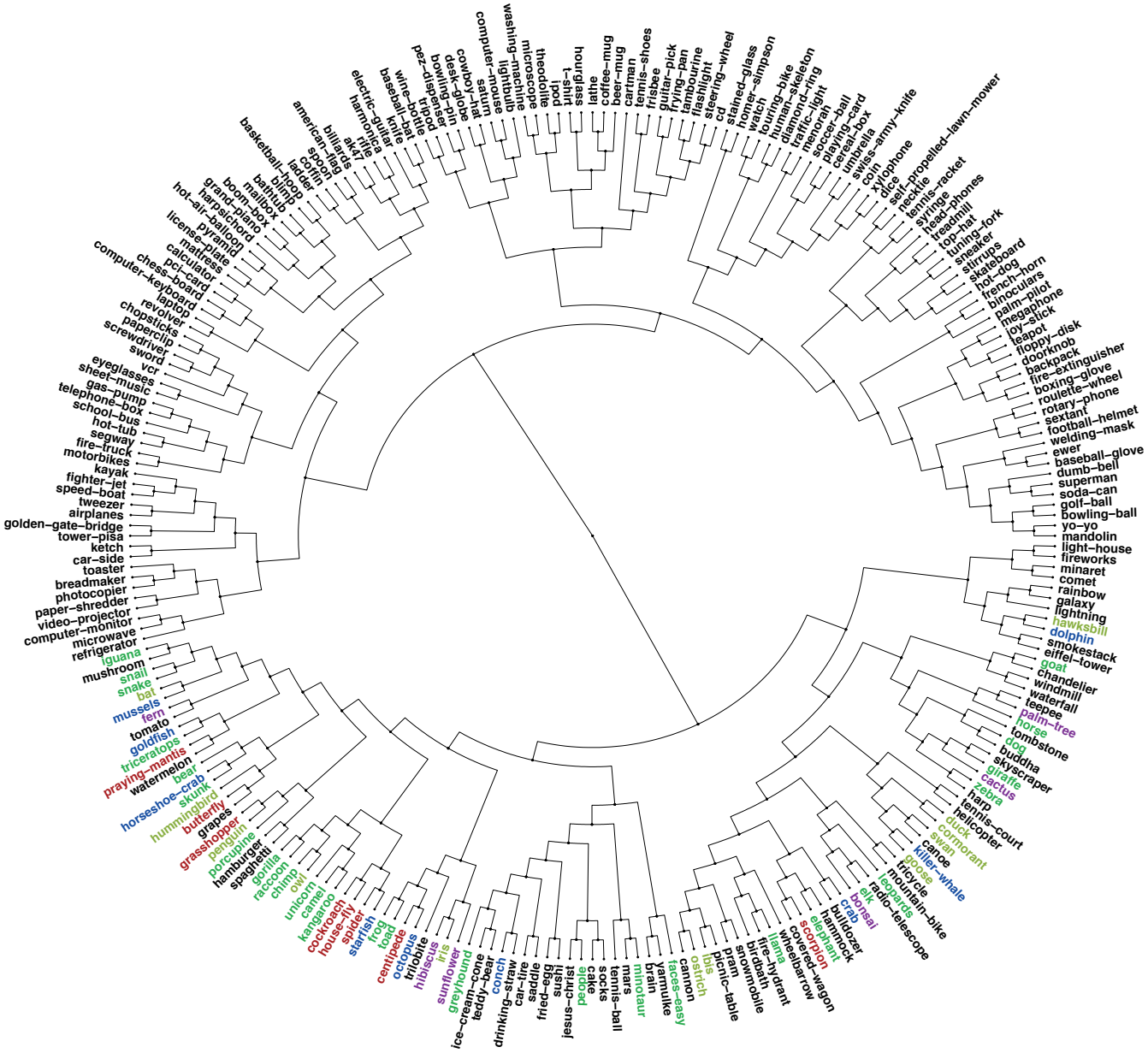

Figure 6. Taxonomy discovered automatically by the computer, using only a limited subset of Caltech-256 training images and their labels. Aside from these labels there is no other human supervision; branch membership is not hand-tuned in any way. The taxonomy is created by first generating a confusion matrix for $N_{\text{train}} = 10$ and recursively dividing it by spectral clustering. Branches and their categories are determined solely on the basis of the confusion between categories, which in turn is based on the feature-matching procedure of Spatial Pyramid Matching. To compare this with some recognizably human categories we color code all the insects (red), birds (yellow), land mammals (green) and aquatic mammals (blue). Notice that the computer's hierarchy usually begins with a split that puts all the plant and animal categories together in one branch. This split is found automatically with such consistency that in a third of all randomized training sets *not a single category of living thing* ends up on the opposite branch.

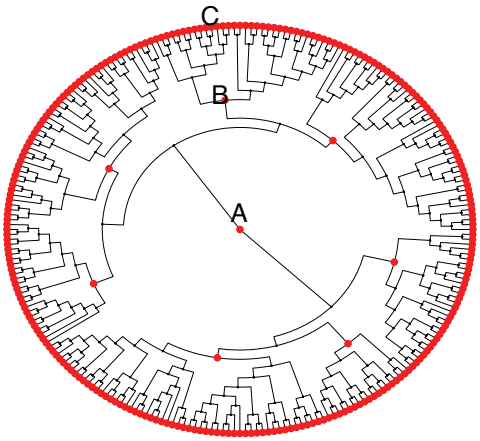

Figure 7. The taxonomy from Fig.6 is reproduced here to illustrate how classification performance can be traded for classification speed. Node *A* represents an ordinary non-hierarchical one-vs-all classifier implemented using an SVM. This is accurate but slow because of the large combined set of support vectors in $N_{cats} = 256$ individual binary classifiers. A the other extreme, each test image passes through a series of inexpensive binary branch classifiers until it reaches 1 of the 256 leaves, collectively labeled *C* above. A compromise solution B invokes a finite set of branch classifiers prior to final multi-class classification in one of 7 terminal nodes.

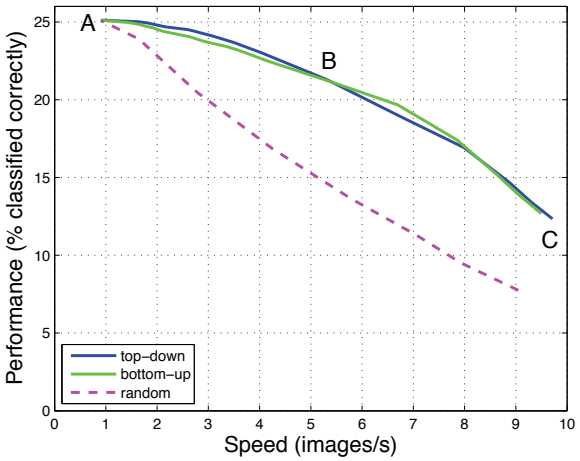

Figure 8. Comparison of three different methods for generating taxonomies. For each taxonomy we vary the number of branch comparisons prior to final classification, as illustrated in Fig. 4. This results in a tradeoff between performance and speed as one moves between two extremes *A* and *C*. Randomly generated hierarchies result in poor cascade performance. Of the three methods, taxonomies based on Spectral Clustering yield marginally better performance. All three curves measure performance vs. speed for $N_{\text{cat}} = 256$ and $N_{\text{train}} = 10$.

2.4. Hierarchical Approach

Our hierarchical classification architecture is shown in Fig. 4. The principle behind the architecture is simple: rather than a single one-vs-all classifier, we achieve classification by recursively splitting the set of possible labels into two roughly equal subsets. This divide-and-conqueor strategy is familiar to anyone who has played the game of 20 questions.

This method is faster because the binary *branch* classifiers are less complex than the one-vs-all *node* classifiers. For example the 1-vs-N node classifier at the top of Fig. 1 actually consists of N=8 separate binary classifiers, each with its own set $\mathcal{S}_{\text{i}}$ of support vectors. During classification each test image must now be compared with the union of training images

$$S_{\text{node}} = \bigcup_{i=1}^{N} \mathcal{S}_{\text{i}}$$

Unless the sets $\mathcal{S}_{\text{i}}$ happen to be the same (which is highly unlikely) the size of S_{node} will increase with N.

Our procedure works as follows. In the first stage of classification, each test image reaches its terminal node via a series of ℓ inexpensive branch comparisons. By the time the test image arrives at its terminal node there are only $\sim N_{\text{cat}}/2^{\ell}$ categories left to consider instead of N_{cat}. The greater the number of levels ℓ in the hierarchy, the fewer categories there are to consider at the expensive final stage - with correspondingly fewer support vectors overall.

The main decision to be taken in building such a hierarchical classification tree is how to choose the sets into which each branch divides the remaining categories. The key intuition which guides our architecture is that decisions between categories that are more easily confused should be taken later in the decision tree, i.e. at the lower nodes where fewer categories are involved. With this in mind we start the training phase by constructing a confusion matrix $\mathcal{C}'_{ij}$ from the training set alone using a leave-one-out validation procedure. This matrix (see Fig. 5) is used to estimate the affinity between categories. This should be distinguished from the standard confusion matrix $\mathcal{C}_{ij}$ which measures the confusion between categories during the *testing* phase.

3. Building Taxonomies

Next, we compare two different methods for generating taxonomies automatically based on the confusion matrix $\mathcal{C}'_{ij}$.

The first method splits the confusion matrix into two groups using Self-Tuning Spectral Clustering [27]. This is a variant of the Spectral Clustering algorithm which automat-

ically chooses an appropriate scale for analysis. Because our cascade is a binary tree we always choose two for the number of clusters. Fig. 4 shows only the first two levels of splits while Fig. 6 repeats the process until the leaves of the tree contain individual categories.

The second method builds the tree from the bottom-up. At each step the two groups of categories with the largest mutual confusion are joined while their confusion matrix rows/columns are averaged. This greedy process continues until there is only a single super-group containing all 256 categories. Finally, we generate a random hierarchy as a control.

4. Top-Down Classification Algorithm

Once a taxonomy of classes is discovered, we now seek to exploit this taxonomy for efficient top-down classification. The problem of multi-stage classification has been studied in many different contexts [1, 6, 17, 16]. For example, Viola and Jones [25] use an attentional cascade to quickly exclude areas of their image that are unlikely to contain a face. Instead of using a tree, however, they use a linear cascade of classifiers that are progressively more complex and computationally intensive. Fleuret and German [9] demonstrate a hierarchy of increasingly descriminative classifiers which detect faces while also estimating pose.

Our strategy is illustrated in Fig. 4 and described in its caption. We represent the taxonomy of categories as a binary tree, taking the two largest branches at the root of the tree and calling these classes $A \cup B$ and $C \cup D$. Now take a random subsample of F_{train} of the training images in each of the two branches and label them as being in either class 1 or 2. An SVM is trained using the Spatial Pyramid Matching kernel as before except that there are now two classes instead of N_{cat}. Empirically we find that $F_{\text{train}} = 10\%$ significantly reduces the number of support vectors in each branch classifier with little or no performance degradation.

If the branch classifier passes a test image down to the left branch, we assume that it cannot belong to any of the classes in the right branch. This continues until the test image arrives at a terminal node. Based on the above assumption, for each node at depth ℓ, the final multi-class classifier can ignore roughly $1-2^{-\ell}$ of the training classes. The exact fraction varies depending on how balanced the tree is.

The overall speed per test image is found by taking a union of all the support vectors required at each level of classification. This includes all the branch and node classifiers which the test image encounters prior to final classification. Each support vector corresponds to a training image whose matching score must be computed, at a cost of 0.4 ms per support vector on a Pentium 3 GHz machine. As already noted, the multi-class node classifiers require many more support vectors than the branch classifiers. Thus increasing the number of branch classifier levels decreases the overall number of support vectors and increases the classification speed, but at a performance cost.

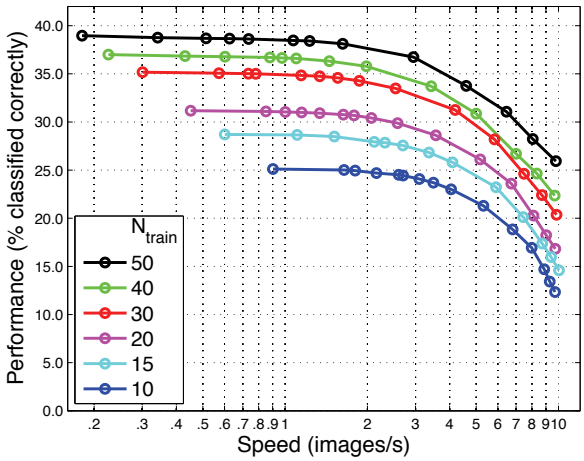

Figure 9. Cascade performance / speed trade-off as a function of N_{train}. Values of $N_{\text{train}} = 10$ and $N_{\text{train}} = 50$ result in a 5-fold and 20-fold speed increase (respectively) for a fixed 10% perforance drop.

5. Results

As shown in Fig. 8, our top-down and bottom-up methods give comparable performance at $N_{\text{train}} = 10$. Classification speed increases 5-fold with a corresponding 10% decrease in performance. In Fig. 9 we try a range of values for N_{train}. At $N_{\text{train}} = 50$ there is a 20-fold speed increase for the same drop in performance.

6. Conclusions

Learning hierarchical relationships between categories of objects is an essential part of how humans understand and analyze the world around them. Someone playing the game of "20 Questions" must make use of some preconceived hierarchy in order to guess the unknown object using the fewest number of queries. Computers face the same dilemma: without some knowledge of the taxonomy of visual categories, classifying thousands of categories is reduced to blind guessing. This becomes prohibitively inefficient as computation time scales linearly with the number of categories.

To break this linear bottleneck, we attack two separate problems. How can computers automatically generate useful taxonomies, and how can these be applied to the task of classification? The first problem is critical. Taxonomies built by hand have been applied to the task of visual classification [29] for a small number of categories, but this method

does not scale well. It would be tedious - if not impossible - for a human operator to generate detailed visual taxonomies for the computer, updating them for each new environment that the computer might encounter. Another problem for this approach is consistency: any two operators are likely to construct entirely different trees. A more consistent approach is to use an existing taxonomy such as WordNet [7] and apply it to the task of visual classification [19]. One caveat is that lexical relationships may not be optimal for certain visual classification tasks. The word *lemon* refers to an unreliable *car*, but the visual categories lemon and car are not at all similar.

Our experiments suggest that plausible taxonomies of object categories can be created automatically using a classifier (in this case, Spatial Pyramid Matching) coupled to a learning phase which estimates inter-category confusion. The only input used for this process is a set of training images and their labels. The taxonomies such the one shown in Fig. 6 seem to consistently discover broader categories which are naturally recognizable to humans, such as the distinction between animate and inanimate objects.

How should we compare one hierarchy to another? It is difficult to quantify such a comparison without a specific goal in mind. To this end we benchmark a cascade of classifiers based on our hierarchy and demonstrate significant speed improvements. In particular, top-down and bottom-up recursive clustering processes both result in better performance than a a randomly generated control tree.

7. Acknowledgments

Research funded by National Science Foundation grant NSF IIS-0535292 and by ONR MURI grant N00014-06-0734.

References

[1] Y. Amit, D. Geman, and X. Fan. A coarse-to-fine strategy for multiclass shape detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(12):1606–1621, 2004.

[2] J. S. Beis and D. G. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *CVPR*, pages 1000–1006, 1997.

[3] I. Biederman. Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94:115–147, 1987.

[4] M. C. Burl and P. Perona. Recognition of planar object classes. In *CVPR*, pages 223–230, 1996.

[5] M. Everingham et al. The 2005 pascal visual object classes challenge. In *MLCW*, pages 117–176, 2005.

[6] X. Fan. Efficient multiclass object detection by a hierarchy of classifiers. In *CVPR (1)*, pages 716–723, 2005.

[7] C. Fellbaum, editor. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. The MIT Press, May 1998.

[8] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *CVPR (2)*, pages 264–271, 2003.

[9] F. Fleuret and D. Geman. Coarse-to-fine face detection. *International Journal of Computer Vision*, 41(1/2):85–107, 2001.

[10] K. Grauman and T. Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *ICCV*, pages 1458–1465, 2005.

[11] K. Grauman and T. Darrell. Unsupervised learning of categories from sets of partially matching image features. In *CVPR (1)*, pages 19–25, 2006.

[12] G. Griffin, A. Holub, and P. Perona. Caltech-256 image dataset. Technical report, Computation and Neural Systems Dept., 2006.

[13] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR (2)*, pages 2169–2178, 2006.

[14] B. Leibe and B. Schiele. Scale-invariant object categorization using a scale-adaptive mean-shift search. In *DAGM-Symposium*, pages 145–153, 2004.

[15] F. Li, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. 2004.

[16] T. Li. Hierarchical document classification using automatically generated hierarchy. In *SDM*, 2005.

[17] T.-Y. Liu, Y. Yang, H. Wan, H.-J. Zeng, Z. Chen, and W.-Y. Ma. Support vector machines classification with a very large-scale taxonomy. *SIGKDD Explorations*, 7(1):36–43, 2005.

[18] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[19] M. Marszałek and C. Schmid. Semantic hierarchies for visual object recognition. In *IEEE Conference on Computer Vision & Pattern Recognition*, jun 2007.

[20] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A Comparison of Affine Region Detectors. *International Journal of Computer Vision*, 2005.

[21] J. Mutch and D. G. Lowe. Multiclass object recognition with sparse, localized features. In *CVPR (1)*, pages 11–18, 2006.

[22] T. Serre, L. Wolf, and T. Poggio. Object recognition with features inspired by visual cortex. In *CVPR (2)*, pages 994–1000, 2005.

[23] M. Varma and D. Ray. Learning the discriminative power-invariance trade-off. In *Proceedings of the IEEE International Conference on Computer Vision, Rio de Janeiro, Brazil*, October 2007.

[24] P. A. Viola and M. J. Jones. Robust real-time face detection. In *ICCV*, page 747, 2001.

[25] P. A. Viola and M. J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[26] M. Weber, M. Welling, and P. Perona. Unsupervised learning of models for recognition. In *ECCV (1)*, pages 18–32, 2000.

[27] L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. *Eighteenth Annual Conference on Neural Information Processing Systems, (NIPS)*, 2004.

[28] H. Zhang, A. C. Berg, M. Maire, and J. Malik. Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In *CVPR (2)*, pages 2126–2136, 2006.

[29] A. Zweig and D. Weinshall. Exploiting object hierarchy: Combining models from different category levels. *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8, 14-21 Oct. 2007.