Constructing Category Hierarchies
for Visual Recognition

Marcin Marszałek and Cordelia Schmid

INRIA Grenoble, LEAR, LJK
{marcin.marszalek,cordelia.schmid}@inrialpes.fr

Abstract. Class hierarchies are commonly used to reduce the complexity of the classification problem. This is crucial when dealing with a large number of categories. In this work, we evaluate class hierarchies currently constructed for visual recognition. We show that top-down as well as bottom-up approaches, which are commonly used to automatically construct hierarchies, incorporate assumptions about the separability of classes. Those assumptions do not hold for visual recognition of a large number of object categories. We therefore propose a modification which is appropriate for most top-down approaches. It allows to construct class hierarchies that postpone decisions in the presence of uncertainty and thus provide higher recognition accuracy. We also compare our method to a one-against-all approach and show how to control the speed-for-accuracy trade-off with our method. For the experimental evaluation, we use the Caltech-256 visual object classes dataset and compare to state-of-the-art methods.

1 Introduction

Visual object classification is one of the basic computer vision problems. In spite of significant research progress, the problem is still far from being solved and a considerable effort is still being put into this research area [1].

In the last years, one could witness remarkable progress in the development of robust image representations and also observe successful applications of sophisticated machine learning techniques in computer vision. Developments in image representation include research on interest point detectors [2, 3], SIFT features [4] and bag-of-features [5]. Support Vector Machines (SVMs) [6] were successfully applied to vision with the design of specialized kernels [7, 8]. Combining these techniques allowed researchers to construct successful visual object recognition systems [1]. We build on those works to construct our baseline.

Still, the typical problems that are tackled today by the state-of-the-art visual object class recognition systems, consist of only few object categories. Very recently, datasets that include more than a hundred of categories, like the most recent Caltech datasets [9, 10], have been introduced. Furthermore, there is an obvious need to further increase this number. In this paper we examine the problem of classifying a large number of categories and use the Caltech-256 [10] dataset for evaluation. Figure 1 shows a few sample images.

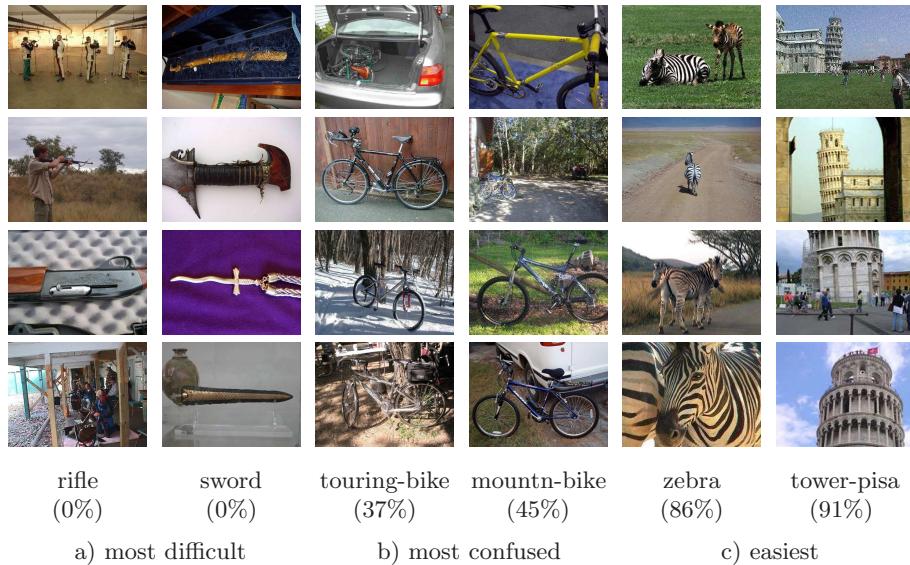| rifle (0%)	sword (0%)	touring-bike (37%)	mountn-bike (45%)	zebra (86%)	tower-pisa (91%)
a) most difficult		b) most confused		c) easiest	

Fig. 1: Sample Caltech-256 images for the most difficult (left), the most confused (middle) and the easiest (right) classes are shown. In parentheses the per-class accuracy of our method is given.

Multi-class classification problems are often handled by combining multiple binary classifiers. Common setups for binary SVMs are based on competition (one-against-rest), voting (one-against-one) or discarding subsequent hypotheses (DAG-SVM). Unfortunately, this means linear (OAR, DAG) or square (OAO) complexity in the number of classes and therefore does not scale well. In principle, a direct multi-class approach is also possible for SVMs, but the optimization problem becomes significantly more difficult and in spite of longer training times, the accuracy is comparable to the one achieved with an OAR approach [6].

To deal with a large number of categories, many approaches combine binary classifiers using class hierarchies. This usually results in logarithmic complexities. The question of how to build such a hierarchy remains, however, open. Approaches common in computer vision can be divided into two groups. First, the hierarchy can be constructed top-down by recursive partitioning of the set of classes. To find a decomposition, Chen et al. [11] used a sophisticated relaxed max-cut formulation, while Liu et al. [12] simply employed k-means clustering. Second, the hierarchy can be built bottom-up by agglomerative clustering. Zhigang et al. [13] explored this approach. In principle, hierarchies could also be found by exhaustive search or random sampling followed by cross-validation. Yuan et al. [14] compared hierarchies found with exhaustive search with the ones constructed by a k-means based top-down method. For a small number of categories, using a top-down method resulted in performance comparable to employing an exhaustive search. For a large number of categories, the expo-

nential growth of possible solutions prohibits the use of exhaustive or random approaches, so we do not include them into our further consideration.

Class hierarchies can address the limitation of current systems to handle a large number of object categories, but they can be used for visual recognition in other ways as well. For example, Zweig and Weinshall [15] exploited class hierarchies to combine models form different category levels, whereas He and Zemel [16] used them to cope with missing and roughly-specified annotations. As the need for class hierarchies increases, the purpose of this paper is to evaluate the suitability of currently constructed hierarchies for visual recognition. We observe that even the most recent methods tend to model class hierarchies with trees [17]. As we will show, this imposes a hard constraint that leads to separation problems when the number of categories increases. We propose a simple yet powerful solution based on the relaxation of this constraint and the possibility of postponing uncertain decisions until they can be reliably made. Furthermore, we address the classification complexity in the number of classes by demonstrating how one can control speed-for-accuracy trade-off with our method.

Hierarchical methods are also used at lower levels. Approaches like vocabulary trees [18] that speed up feature matching are related to our work due to their hierarchical nature. Similarly, kd-trees are hierarchic space partitioning structures that can perform component-wise classification [19]. Note, however, that in this work we focus on building high-level class hierarchies and look into the problem of class-wise partitioning.

The rest of the paper is organized as follows. In Sect. 2 we evaluate existing approaches for constructing class hierarchies. In Sect. 3 we propose a novel approach that avoids the separation problem present in existing methods. In Sect. 4 we experimentally confirm our findings and demonstrate the speed-for-accuracy trade-off of our method. We conclude the paper in Sect. 5.

2 Existing Approaches

In this section we assume that some dissimilarity measure between classes is given. Common approaches are to simply compute the distances between class means [12, 14] or to represent the classes in a high dimensional space using a Gaussian kernel [13, 20].

As discussed in the introduction, we can divide the existing methods for building class hierarchies into two main groups. In Subsect. 2.1 we consider some commonly used methods that construct the class hierarchy by top-down recursive partitioning of the set of classes. In Subsect. 2.2 we discuss methods based on bottom-up agglomerative clustering of the classes.

Given a class hierarchy, we can efficiently classify samples by descending the resulting decision tree. In principle, any classifier could be used in the nodes of the hierarchy to make the decision about the direction of descent. In practice, Support Vector Machines are widely used for this task in computer vision.

Most often the hierarchies are represented as binary trees, which means that at each node a binary decision is made on which of the two subtrees to choose.

Thus, a Support Vector Machine for each node of the tree can be trained. If the tree is balanced, only $\lceil log_2 N \rceil$ SVM runs are necessary to perform the N-class classification. In the worst case (degenerated trees) the complexity is linear in the number of classes. Therefore, in general, hierarchy-based classification approaches scale well with the number of classes.

2.1 Top-Down Recursive Partitioning

K-means clustering. A set of classes can be clustered into k groups. This determines the partitioning at a given level. When applied recursively, this simple yet popular [12, 14, 20] method allows to construct a class hierarchy. K-means clustering minimizes the distances to cluster centers, thus tries to find compact clusters. This presumably leads to well separated clusters.

Normalized cuts. A dataset can be viewed as a fully connected undirected graph $G_V = (V, E_V)$, where $v \in V$ nodes correspond to the elements of the dataset and edge weights $k(v_1, v_2) \in E_V$ correspond to the similarity measure between the elements. This is the starting point of many spectral clustering algorithms. A *graph cut* partitions G_V into G_A and G_B, where $A \subset V$ and $B \subset V$ are two disjoint sets of nodes $A \sqcup B = V$, i.e., $A \cup B = V$ and $A \cap B = \emptyset$. Shi and Malik [21] proposed the Normalized Cuts method to find a good cut through such a graph. The idea is to minimize

$$\text{Ncut}(A, B) = \frac{\text{assoc}(A, B)}{\text{assoc}(A, V)} + \frac{\text{assoc}(A, B)}{\text{assoc}(B, V)} \qquad (1)$$

where $\text{assoc}(A, B)$ is the weight of all edges connecting the nodes between the sets A and B, i.e.,

$$\text{assoc}(A, B) = \sum_{a \in A, b \in B} k(a, b) \ . \qquad (2)$$

Note that $\text{assoc}(A, B)$ is often denoted in the literature as $\text{cut}(A, B)$. As the distance measures used in spectral clustering are often positive definite, the adjacency matrix E_V is often denoted as K. The common choice is the RBF kernel, which can be generalized to an extended Gaussian kernel

$$k(v_i, v_j) = K_{ij} = e^{-\frac{1}{\gamma} m(v_i, v_j)} \qquad (3)$$

where $m(v_i, v_j)$ is the distance measure between the elements.

Finding the optimal normalized cut is NP-hard, therefore the following relaxation is commonly used:

$$w^* = \arg\max_w \frac{w^T D^{-\frac{1}{2}} K D^{-\frac{1}{2}} w}{w^T w} \tag{4}$$

such that

$$w^T D \mathbf{1} = 0 \tag{5}$$

where D is a diagonal matrix with $d_{ii} = \sum_j K_{ij}$ and $\mathbf{1}$ is the vector of all ones. The optimal w^* can be found by computing the eigenvector corresponding to the second largest eigenvalue of $D^{-\frac{1}{2}} K D^{-\frac{1}{2}}$. The $\text{sgn}(w_i^*)$ indicates whether $v_i \in A$ or $v_i \in B$. As it was shown by Rahimi and Recht [22], this relaxation can be interpreted as finding a maximal hyperplanar gap.

By recursively partitioning the graph G_V where V is a set of classes, one can obtain a class hierarchy [11].

2.2 Bottom-Up Agglomerative Clustering

Given a distance measure between classes, agglomerative clustering can be used to build a class hierarchy bottom up. Initially, all the classes belong to different clusters. Then, the closest clusters are merged. It is assumed that merging close elements will result in clusters that are better separated. The distances to the new-formed class clusters are recomputed, such that the procedure can be applied iteratively until all classes belong to one cluster. The merge order determines the hierarchy. If during each step one looks for the two most similar clusters, the hierarchy will have a form of a binary tree. Zhigang et al. [13] explored both binary trees and k-trees.

2.3 Discussion

Most existing class hierarchy construction methods assume that at each level of the hierarchy the feature-space can be partitioned into disjoint subspaces. We predict an inevitable conflict between generalization and precision requirements. Especially for the earliest decisions, where the boundary is supposed to split very distinct categories of objects (natural vs. man-made objects for example), a requirement is enforced to precisely trace the boundaries between tens or hundreds of similar classes that fall at the explored decision boundary (a bear vs. a teddy-bear and a fountain vs. waterfall for example). Note that a mistake at a boundary of such a high-level decision is as costly as a mistake at lower levels, where the classifier can tune to minor class differences without degrading its generalization properties.

Given a few distinct visual object categories class separability can be good. But this certainly cannot hold for hundreds or thousands of classes. Let us motivate our hypothesis with some simplified examples before evaluating it experimentally.
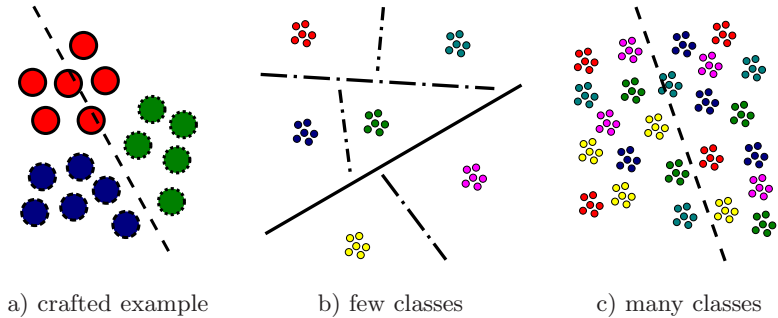
a) crafted example b) few classes c) many classes

Fig. 2: Simple examples of separating 2-dimensional multi-class data with a linear decision boundary. Difficulties to separate classes (left) might not arise for a few separated classes (middle), but can emerge when the number of classes increases (right).

Figure 2 presents some simplistic efforts to separate 2-dimensional multi-class data with a linear boundary. A carefully crafted example (Fig. 2a) shows, that even if any two of three classes can be easily separated with a hyperplane, it does not assure good separation of all three classes. If there are few classes which are well separated (Fig. 2b), a good recursive partitioning can be found. With the growing number of classes, however, it will be increasingly difficult to find a disjoint class-set partitioning (Fig. 2c).

As we show in Sect. 4, early enforcement of hard decisions can be costly in the hierarchic setup and can significantly lower the classification performance. Thus, we propose a novel approach for constructing top-down hierarchies, which postpones final decisions in the presence of uncertainty.

3   Our Approach

Our approach is based on the observation that finding a feature-space partitioning that reflects the class-set partitioning becomes more and more difficult with a growing number of classes. Thus, we propose to avoid disjoint partitioning and split the class-set into overlapping sets instead. This allows to postpone uncertain classification decisions until the number of classes gets reduced and learning good decision boundaries becomes tractable.

The proposed solution is to discover classes that lie on the partition boundary and could introduce classification errors. Those classes should not be forced into either of the partitions, but they should be included in both. With our approach, a number of classes can still be separated with one decision. This assures a computational gain compared to setups with linear complexity like OAR. However, since disjoint partitioning is not enforced, the performance is not degraded. As the resulting partitioning is relaxed, we call our hierarchy Relaxed Hierarchy (RH).

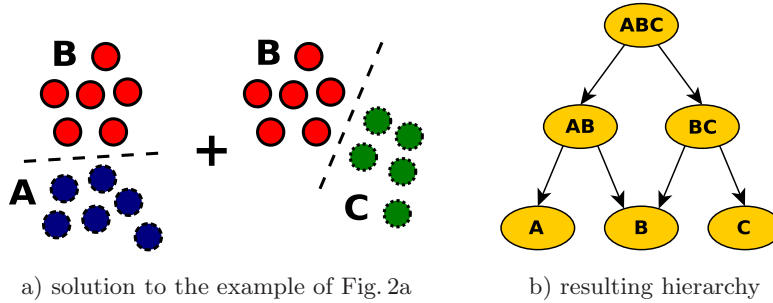a) solution to the example of Fig. 2a   b) resulting hierarchy

Fig. 3: Illustration of our approach. Separation difficulties can be avoided by including non-separable classes on both sides of the decision boundary. This can simplify subsequent splits (left) and leads to a rooted DAG structure (right).

Figure 3 demonstrates how our method applies to the problem sketched in Subsect. 2.3. The boundary from Fig. 2a which separates members of a class can be used if both subpartitions (Fig. 3a) contain this class. Moreover, the subsequent splits are straightforward. Note that the resulting hierarchy (Fig. 3b) is no longer a tree, but a rooted directed acyclic graph (DAG).

Our method can be applied to most top-down partitioning approaches. This includes methods based on k-means clustering and normalized cuts. Here we build on normalized cuts. Note that the kernel matrix constructed for SVMs can be reused. Furthermore, only one eigenvector corresponding to the second largest eigenvalue needs to be computed, so optimized algorithms can be used.

By partitioning the set of training samples S instead of the set of classes $\mathcal{C} = \{[s] : s \in S\}$,[1] a separating boundary between the samples can be found. A disjoint bi-partitioning of samples $S = A \sqcup B$ leads to a disjoint tri-partitioning of classes $\mathcal{C} = \mathcal{A} \sqcup \mathcal{X} \sqcup \mathcal{B}$, where all classes in $\mathcal{A}$ have all samples in A, all classes in $\mathcal{B}$ have all samples in B, and finally the rest of the classes $\mathcal{X}$ have samples in both partitions. Our proposal is to split the set of classes $\mathcal{C} = \mathcal{L} \cup \mathcal{R}$ so that the classes in $\mathcal{X}$ belong to both sets, i.e., $\mathcal{X} = \mathcal{L} \cap \mathcal{R}$:

$$\mathcal{L} = \mathcal{A} \cup \mathcal{X} = \{C : \exists_{s \in A} [s] = C\}$$
$$\mathcal{R} = \mathcal{B} \cup \mathcal{X} = \{C : \exists_{s \in B} [s] = C\} \ . \tag{6}$$

In practice, we can also slightly relax the requirement for $\mathcal{A}$ ($\mathcal{B}$) to have all samples in A (B). Given a partitioning $p : S \to \{-1, 1\}$ of the training set S, we define a function $q : \mathcal{C} \to [-1, 1]$ on the set of classes $\mathcal{C}$:

$$q(C) = \frac{1}{|C|} \sum_{s \in C} p(s) \tag{7}$$

where $C \in \mathcal{C}$ is a class.

---

[1] $[s]$ denotes the class assigned to sample $s \in S$
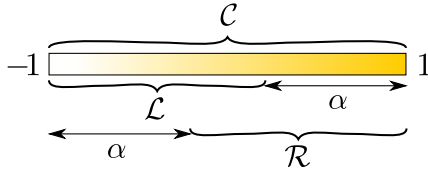
Fig. 4: Illustration of the split procedure. Note how the value of α influences the overlap.

This allows us to define a split:

$$\begin{aligned}\mathcal{L} &= q^{-1}\big([-1, 1-\alpha)\big) \\ \mathcal{R} &= q^{-1}\big((-1+\alpha, 1]\big)\end{aligned} \tag{8}$$

where q^{-1} denotes an inverse image and α is a softening parameter. Note that in this case $\mathcal{A} = q^{-1}\big([-1, -1+\alpha]\big)$, $\mathcal{B} = q^{-1}\big([1-\alpha, 1]\big)$ and $\mathcal{X} = q^{-1}\big((-1+\alpha, 1-\alpha)\big)$, thus when $\alpha = 0$ then the above simplifies to (6).

Figure 4 illustrates the split procedure. Let us consider a set of classes $\mathcal{C}$ ordered according to $q(C)$ values for each class $C \in \mathcal{C}$. The $\mathcal{L}$ set consists of classes $L \in \mathcal{L}$ with $q(L) < 1 - \alpha$ and the $\mathcal{R}$ set of classes $R \in \mathcal{R}$ with $q(R) > -1 + \alpha$. Increasing α reduces the split overlap. This produces more efficient hierarchies, but can degrade performance.

We create our Relaxed Hierarchy by recursively splitting the class-sets $\mathcal{C}_n$ until they contain only one class or a group of classes impossible to split, i.e., until $|\mathcal{C}_n| = 1$ or $\mathcal{L}_n = \mathcal{R}_n$. In the second case we use OAR on the subset of classes that is too complex to split.

To train the hierarchy, for each node of the computed rooted DAG we train an SVM using samples belonging to classes in $\mathcal{R}_n \setminus \mathcal{L}_n$ as a positive set and to classes in $\mathcal{L}_n \setminus \mathcal{R}_n$ as a negative set. Note that samples belonging to classes in $\mathcal{X}_n = \mathcal{L}_n \cap \mathcal{R}_n$ are not used for training. This does not matter, since classification of a sample that belongs to a class in $\mathcal{X}_n$ is not relevant at this stage. This is the key point of our method, since the decision for these classes could be erroneous and is postponed till later.

For testing, the tree is descended until a leaf is reached. The decision is either directly the class label (leaves containing only one class) or OAR classification is performed on the remaining classes (complex leaves with more that one class).

4   Experiments

In Subsect. 4.1 we describe the implementation details of our image representation and the classifier used. Note, however, that different image representations and classifiers can be combined with our Relaxed Hierarchy. Subsection 4.2 introduces the dataset and the experimental setup. Results are presented and discussed in Subsect. 4.3.

4.1 Image Representation and Image Dissimilarity Measure

Given an image, we use complementary Harris-Laplace [2] and Laplacian interest point detectors [3] to find a sparse set of salient image regions. Both detectors are invariant to scale transformations, they output circular regions at a characteristic scale. Using the SIFT [4] descriptor, gradient orientation histograms are computed over those regions.

To describe an image, we use the bag-of-features representation [5]. Given a visual vocabulary, we represent the appearance of the image as a histogram of vocabulary words occurring in the image. Each histogram entry $h_{ij} \in H_i$ is the proportion of all image features i assigned to a vocabulary word j with respect to the total number of features in the image.

To compute the dissimilarity between the images, we use the χ^2 distance

$$m(H_i, H_j) = \frac{1}{2} \sum_{n=1}^{V} \frac{(h_{in} - h_{jn})^2}{h_{in} + h_{jn}} \quad . \tag{9}$$

where V is the vocabulary size. We use k-means to construct the vocabulary and $V = 8000$ in our experiments.

To use this distance measure in Support Vector Machines, we use the extended Gaussian kernel, cf. (3). This results in a Mercer kernel [23]. The parameter γ is set to the mean value of the distances between all training samples.

Using the above image representation with Support Vector Machines in the OAR setup corresponds to the method of Zhang et al. [8]. This method has shown an excellent performance on varying object class datasets, including 2005 and 2006 PASCAL VOC challenges [8, 24]. Extended with additional channels and a separate optimization framework to combine them, this approach won the PASCAL VOC classification challenge in 2007 [1].

4.2 Caltech-256

We evaluate our approach on the Caltech-256 [10] dataset. It contains images of 256 object categories and an additional background class. Each category contains at least 80 images of varying size and quality. The images were downloaded from the web with the help of popular image search engines and then human-filtered.

We closely follow the experimental setup suggested by the dataset authors, i.e., we use the first 250 categories of the dataset to measure the accuracy of multi-class image classification. The first 15 images of each class are used for training (this includes computing the visual vocabulary for the bag-of-features, constructing the class hierarchy and training the SVMs) and all the remaining images are used for testing. We report the average of the per-class classification accuracy.

Figure 1 shows a few samples and the classification accuracies for the corresponding classes. We can see that classes with very high intra-class variability (like rifles and swords) are the most difficult. Our method also confuses two very similar bike classes (34% touring bikes are classified as mountain bikes and 24%

mountain bikes as touring bikes). It performs very well on classes with discriminative texture (like zebras) and those with low intra-class variability (like the tower of Pisa).

4.3 Results

Figure 5 shows a class hierarchy constructed by our method for the Caltech-256 dataset, displayed for a subset of 10 categories. The categories were chosen to include animals, natural phenomena and man-made objects. They include class pairs with apparent visual similarities that are semantically close (bear and dog, top hat and cowboy hat) as well as those that have a secondary or no semantic relationship at all (bear and teddy bear, top hat and Saturn). The hierarchy reveals many intuitive relationships and groupings. At the top node man-made objects and natural phenomena (hats, lightning, rainbow, Saturn) are separated from animals (octopus, starfish, bear). Classes at the partition boundary (dog and teddy bear) are included in both partitions. Subsequent splits further separate sea animals from land animals (with a teddy bear) and hat-like objects (including Saturn) from natural phenomena and mascot-like objects. Even though it is based on visual data only, the constructed hierarchy turns out to be similar to hierarchies extracted from semantic networks [25]. Unlike the purely semantic hierarchies, however, it also groups classes that are related by semantic links difficult to model (bear and teddy bear) or that feature accidental similarity (top hat and Saturn).

Table 1 shows the average per-class classification accuracy on the Caltech-256 dataset. The upper half of the table compares our approach, i.e., a Relaxed Hierarchy (RH), to the OAR setup. We can see that the proposed hierarchy does not lead to accuracy loss. The image representation is the one described in
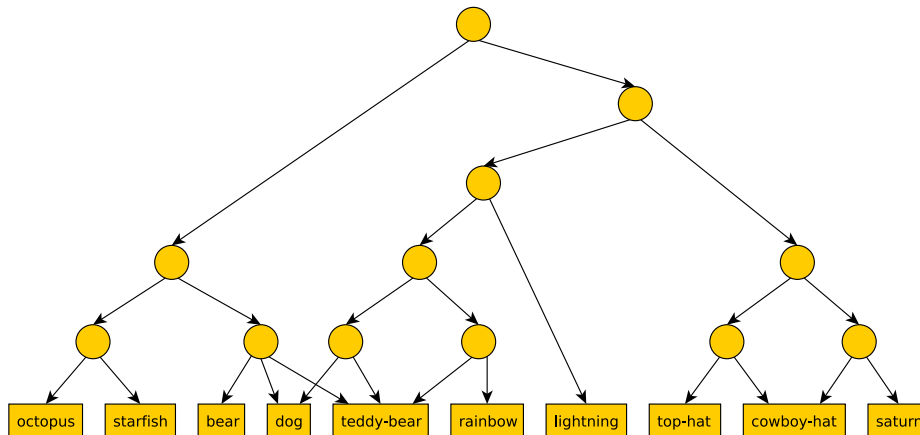

Fig. 5: Class hierarchy constructed by our method for the Caltech-256 dataset, displayed for a subset of 10 categories.

Table 1: Average per-class classification accuracy on the Caltech-256 dataset.

OAR (reimpl. of Zhang et al. [8])	23.6%
Our RH ($\alpha = 0$, sparse IPs)	23.4%
Griffin [10] (reimpl. of Lazebnik at al. [7])	28%
Our RH ($\alpha = 0$, dense/grid)	27.9%

Subsection 4.1. The lower half of the table shows a result for a different image representation, i.e., based on a reimplementation of the method of Lazebnik at al. [7]. This representation obtains better results for the Caltech-256 dataset, as most objects are centered in the image and relatively small. Again, we can observe that the results obtained with our RH and an OAR approach (see results obtained by Griffin et al. [10]) are comparable.

As to be expected, our approach does not depend on the image representation. Best results on Caltech-256 dataset in a similar setup (53% average accuracy for 10 training images) where achieved by Varma [26] using a combination of multiple channels. Our method could be combined with this multi-representation approach. Note that it could even be applied to different data types, but this is beyond the scope of this paper. In the following we use the image representation described in Sect. 4.1 as it is fast to compute and does not impact the evaluation of our class hierarchy construction.

Figure 6 compares the complexity in the number of categories. The complexity in the OAR setup is linear (red squares). The complexity of our Relaxed Hierarchy method is confirmed to be sublinear. The exact gain depends on the
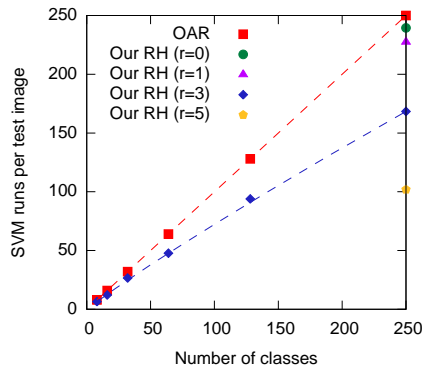

Fig. 6: Complexity in the number of classes. The α relaxation parameter expressed in the number of per-class training samples r (i.e., $\alpha = r/15$) is given in parenthesis for our method.
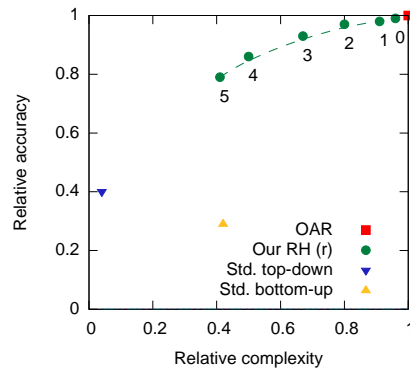
Fig. 7: Speed-for-accuracy trade-off and comparison with existing approaches. Next to the RH datapoints the α relaxation parameter expressed in the number of samples r is shown ($\alpha = r/15$).

parameter α, see the datapoints along the right edge. Note that α is expressed here as r—the number of relaxed training samples per class, i.e., $\alpha = r/15$. For 250 categories and a setting of $\alpha = 3/15 = 0.2$ (blue diamonds) which corresponds to minor performance loss, we observe a reduction of the computation time by $1/3$. This ratio will further increase with the number of categories.

Figure 7 demonstrates the speed-for-accuracy trade-off (green circles) that can be tuned with the α parameter. As shown in Sect. 3, with the increase of the parameter value the set of classes is more willingly treated as separable. Greater α values lead to better computational gain, but could degrade the classification accuracy. Note that the complexity is sublinear independently of the parameter setting (see Fig. 6), but for the smaller number of classes one may choose to accept a small loss in accuracy for a significant gain in computation time. For instance, for Caltech-256 we find the setting of $\alpha = 0.2$ ($r = 3$) reasonable, as the absolute loss in the accuracy is only about 2%, while the computational gain of $1/3$ is noticeable. Setting $\alpha = 0.33$ ($r = 5$) leads to the computational gain of $3/5$, but in exchange for another 2% of accuracy.

Figure 7 compares the results obtained with our class hierarchies for different α values (green circles) to two existing methods for class hierarchy construction (triangles). The baseline top-down method follows the approach of Liu et al. [12, 14], but we use normalized cuts instead of k-means. This makes it more comparable to our method and is also similar to the approach of Chen et al. [11]. The baseline bottom-up method follows the agglomerative clustering based approach of Zhigang et al. [13], but uses the same inter-class similarity measure as the top-down approach. Note that the only difference between the compared methods is the algorithm used for class hierarchy construction, i.e., we keep the same image representation and settings of the Support Vector Machines. Still, the magnitude of the difference is surprising. The standard bottom-up method seems to fail completely. The standard top-down approach has low computational complexity, but the loss in terms of classification accuracy is enormous. This confirms our claim, see Subsect. 2.3, that popular *disjoint* approaches for construction of class hierarchies fail when dealing with a large number of visual object categories. Note that the cited methods were evaluated on visual data and performed well. However, the number of categories never exceeded 14 classes and was usually kept below 10.

5   Summary

We have shown that existing approaches for constructing class hierarchies for visual recognition do not scale well with the number of categories. Methods that perform disjoint class-set partitioning assume good class separability and thus fail to achieve good performance on visual data when the number of categories becomes large. Thus, we have proposed a method that detects classes at the partitioning boundary and postpones uncertain decisions until the number of classes becomes smaller. Experimental validation shows that our method is sublinear in the number of classes and its classification accuracy is comparable to the OAR

setup. Furthermore, our approach allows to tune the speed-for-accuracy trade-off and, therefore, allows to significantly reduce the computational costs.

Our method finds a reliable partitioning of the categories, but the hierarchy may be far from optimal. Finding the optimal partitioning is a hard problem. For the future work we plan use semantic information to drive the optimization.

References

1. Everingham, M., van Gool, L., Williams, C., Winn, J., Zisserman, A.: Overview and results of classification challenge. The PASCAL VOC'07 Challenge Workshop, in conj. with ICCV (2007)
2. Mikolajczyk, K., Schmid, C.: Scale and affine invariant interest point detectors. IJCV (2004)
3. Lindeberg, T.: Feature detection with automatic scale selection. IJCV (1998)
4. Lowe, D.: Distinctive image features form scale-invariant keypoints. IJCV (2004)
5. Willamowski, J., Arregui, D., Csurka, G., Dance, C.R., Fan, L.: Categorizing nine visual classes using local appearance descriptors. In: IWLAVS. (2004)
6. Schölkopf, B., Smola, A.: Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond. (2002)
7. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: CVPR. (2006)
8. Zhang, J., Marszalek, M., Lazebnik, S., Schmid, C.: Local features and kernels for classification of texture and object categories: A comprehensive study. IJCV (2007)
9. Fei-Fei, L., Fergus, R., Perona, P.: One-shot learning of object categories. PAMI (2007)
10. Griffin, G., Holub, A., Perona, P.: Caltech-256 object category dataset. Technical report (2007)
11. Chen, Y., Crawford, M., Ghosh, J.: Integrating support vector machines in a hierarchical output space decomposition framework. In: IGARSS. (2004)
12. Liu, S., Yi, H., Chia, L.T., Deepu, R.: Adaptive hierarchical multi-class SVM classifier for texture-based image classification. In: ICME. (2005)
13. Zhigang, L., Wenzhong, S., Qianqing, Q., Xiaowen, L., Donghui, X.: Hierarchical support vector machines. In: IGARSS. (2005)
14. Yuan, X., Lai, W., Mei, T., Hua, X., Wu, X., Li, S.: Automatic video genre categorization using hierarchical SVM. In: ICIP. (2006)
15. Zweig, A., Weinshall, D.: Exploiting object hierarchy: Combining models from different category levels. In: ICCV. (2007)
16. He, X., Zemel, R.: Latent topic random fields: Learning using a taxonomy of labels. In: CVPR. (2008)
17. Griffin, G., Perona, P.: Learning and using taxonomies for fast visual category recognition. In: CVPR. (2008)
18. Nister, D., Stewenius, H.: Scalable recognition with a vocabulary tree. In: CVPR. (2006)

19. Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.: Object retrieval with large vocabularies and fast spatial matching. In: CVPR. (2007)
20. Casasent, D., Wang, Y.C.: A hierarchical classifier using new support vector machines for automatic target recognition. NN (2005)
21. Shi, J., Malik, J.: Normalized cuts and image segmentation. PAMI (2000)
22. Rahimi, A., Recht, B.: Clustering with normalized cuts is clustering with a hyperplane. In: SLCV. (2004)
23. Fowlkes, C., Belongie, S., Chung, F., Malik, J.: Spectral grouping using the Nyström method. PAMI (2004)
24. Everingham, M., Zisserman, A., Williams, C., van Gool, L.: The PASCAL visual object classes challenge 2006 (VOC2006) results. Technical report (2006)
25. Marszałek, M., Schmid, C.: Semantic hierarchies for visual object recognition. In: CVPR. (2007)
26. Perona, P., Griffin, G., Spain, M. The Caltech 256 Workshop, in conj. with ICCV (2007)