Scalable Multi-class Object Detection

Nima Razavi[1] Juergen Gall[1] Luc Van Gool[1,2]
[1]ETH Zurich, Switzerland [2]KU Leuven, Belgium
{nrazavi,gall}@vision.ee.ethz.ch vangool@esat.kuleuven.be

Abstract

Scalability of object detectors with respect to the number of classes is a very important issue for applications where many object classes need to be detected. While combining single-class detectors yields a linear complexity for testing, multi-class detectors that localize all objects at once come often at the cost of a reduced detection accuracy. In this work, we present a scalable multi-class detection algorithm which scales sublinearly with the number of classes without compromising accuracy. To this end, a shared discriminative codebook of feature appearances is jointly trained for all classes and detection is also performed for all classes jointly. Based on the learned sharing distributions of features among classes, we build a taxonomy of object classes. The taxonomy is then exploited to further reduce the cost of multi-class object detection. Our method has linear training *and* sublinear detection *complexity in the number of classes. We have evaluated our method on the challenging PASCAL VOC'06 and PASCAL VOC'07 datasets and show that scaling the system does not lead to a loss in accuracy.*

1. Introduction

Performance of object detection methods has improved substantially in the recent years [7]. As the number of detectable object classes increases, so does the need for scalable methods for detection. However, scaling multi-class object detection remains a challenging task. Object categories can have a very large intra-class variability in appearance and shape. In a multi-class setup, this poses a challenge since the detection method should be discriminant to inter-class and clutter variations and invariant to intra-class variations.

In order to deal with these issues, several successful methods model the object class by a set of parts[1] and combine them using a shape model [18, 11]. In this scenario, every part has an *appearance* and a *location*. The location is generally defined as the relative position and scale of a feature with respect to a reference point of the object, *e.g.*, center of mass. When extending the part-based approaches to multi-class problems, the parts (or the combination of them) need to be discriminant between the classes.

Figure 1. Features of different object classes can share appearance although they do not necessarily also share their location. For instance, the legs of a person and a horse share both appearance *(bounding boxes)* and location *(arrows)* whereas the wheels of a bus and car are similar in appearance but not in location *(red/blue arrows)*. In this paper, we are building a multi-class detection algorithm by using discriminative features.

State-of-the-art approaches [28, 13] have focused on the appearance of the features for discrimination between classes. In [28], a set of features at fixed locations are selected using a boosting procedure to discriminate objects from clutter and classes from each other. Without enforcing sharing, this procedure ends up at very specific discriminative features which are not generalizing well. Hence, the authors have proposed a solution to this problem by enforcing sharing of features across classes. The selected features, however, become very generic since the location is fixed. These generic features are generalizing well but they are very weakly discriminating between classes and therefore detection with them requires evaluating a model for every class which scales linearly in the number of classes [27]. Similarly in [13], a set of shape features are learned by combining generic simple edge features. However, since the location is fixed in the constellation model, the model of every class needs to be evaluated [12].

In this paper, we propose a method for learning discriminative parts in appearance without fixing their locations. This is achieved by treating location and appearance differently. Our approach is also motivated by psychological ev-

[1]In this paper, the words part and feature are used interchangeably.

idence suggesting that the localization and classification of objects follow different pathways in the human visual system [21, 30]. Such separation implicates also possible computational advantages as discussed in [25]. In our approach, we focus on the features of intermediate complexity which are optimal for classification [29]. In particular, the appearance of our features is shared across categories providing good generalization, yet remaining discriminative for a subset of classes. However, when the appearance is combined with location, the features become discriminative for the individual classes. For instance in Fig. 1, the wheel discriminates the bus from the horse and the person, but not from the car. However, the wheel of the bus gives another detection of the object than the wheel of the car. Hence, we first classify features and obtain a set of likely categories and then do localization to gather evidence for the position of the most likely object class.

For building a shared codebook, we extend the single-class approach of [14] and introduce a novel optimality criteria to achieve the right balance between feature sharing and discrimination in the context of multi-class detection. The proposed multi-class detector scales better than a combination of single-class detectors and, most importantly, is similar in detection accuracy. The detection is based on the implicit shape model [18] where codebook entries vote for the object position and class. Due to the sharing of features, the number of votes that need to be cast for detection also increases only sublinearly with respect to the number of classes. Furthermore, we build a taxonomy of object classes from the sharing distribution among classes. We show that the derived taxonomies have a semantic interpretation and that the taxonomies increase scalability by reducing the number of votes. Since the detection might result in overlapping bounding boxes that are ambiguous in class label, we perform an additional classification after detection. The detection reduces not only the number of potential bounding boxes to a very small number, it also provides a class label such that each bounding box is only classified with respect to a single object class and not all classes.

2. Related Work

Several approaches have addressed the problem of feature sharing in the context of multi-view or multi-class object detection [28, 13, 24, 26]. Following a sliding window approach for detection, [28] proposes a boosting procedure to explicitly enforce sharing and shows that the number of features grows sublinearly with the number of classes. The classification in [28] is done in a one-vs-the-rest approach and scales linearly with the number of classes though. [27] has reduced joint-boost recognition at classification time to nearest neighbor search in a vector space to scale the joint boosting for large multi-class problems. However, this work is limited to the joint-boosting and requires many similar classes. For shape-based object detection, [13] introduces a method for learning a scalable hierarchy of parts by using the statistical co-occurrence of generic features and independently evaluates a constellation model of each class for detection. This work has been recently extended [12] for speeding up multi-class detection by introducing a coarse-to-fine representation of contour features and constellation models. In [28, 12], it is assumed that constellations of similar classes are similar and can therefore be grouped together. However, this assumption is too restrictive as it forces the features in the constellation to simultaneously share appearance and location. As a result, this method yields shallow hierarchies with many disjoint groups of classes and requires again coarsening of features which are once joined in training to increase discriminativity.

Hierarchical taxonomies of object categories have been used previously for scalable image classification [16, 20, 10]. In [16], a taxonomy is built by clustering the confusion matrix of one-vs-the-rest detectors. At each node of the hierarchy, a classifier is trained and combined to a multi-class classifier. The approach has been extended in [20] by allowing overlapping labels in disjoint branches of the hierarchy. The degree of overlap gives a trade-off between accuracy and efficiency. Object hiearchies have been used in [10] to transfer the label into an unseen class by sharing labels between semantically similar classes. Although these works are using hierarchies and are therefore related, they are not applicable to object detection.

Building a codebook of parts and learning appropriate weights for them is also addressed in the literature. [18, 13] follow a generative approach for clustering the patches whereas [26, 19, 31, 14] pursue a discriminative one. [19, 31] build the codebook in a generative way but learn appropriate weights for them discriminatively using a max-margin framework. [26, 14] learn a direct mapping from the patch appearances to weights in a random decision forests framework. In [26], only class labels are used for discriminative training where both class labels and spatial location of features are used in [14]. In [22], scale-invariant features of training data are stored as the codebook without any quantization [2] and used to cast voting lines for detection. Since our features are not scale-invariant, [22] does not directly apply to our approach.

Recently, several approaches have considered generic object detection [1, 6, 3]. These approaches are related to this work as they also separate the localization and classification. In contrast to our work, they do not make use of class information during detection. This implies that detected bounding boxes need to be evaluated for all classes yielding a linear complexity in the number of classes. In this work, we use the label information encoded in the detection for scalable multi-class detection where we do not need to run the final *expensive* classifier for all classes.

3. Analysis of a Multi-class Hough Transform

Let us first introduce some notations. We denote an image by I and the number of pixels in it by $|I|$. Each image is described by a set of features $F^I = \{f_i\}$. Each feature $f_i = (a_i, l_i, c_i)$ has an appearance a_i, location l_i (which fully describes the spatial extent of the feature and is not limited to position necessarily), and belongs to an object from a certain class c_i. The set of all classes is denoted by $C = \{c_1, \ldots, c_n, c_{bg}\}$, where $c_1, \ldots, c_n$ are the positive classes, and c_{bg} is the background class, which needs special treatment as we show later. This way, we can represent an object O by a set of features $F^O = \{a_i^o, l_i^o, c^o\}$ with identical class labels. The locations are represented relative to a reference point, *e.g.*, arithmetic mean of locations. The c^o are known only for training features F^{train}.

A hypothesis h for an object can be represented by a set of features, *i.e.*, $h = \{f_i^h(a_i^h, l_i^h, c_h)\}$, and the scoring function for a hypothesis is denoted by $M : \mathcal{H} \mapsto \mathbb{R}$. The objective of training is to learn a scoring function such that objects get a higher score than non-object hypotheses. Several models have been proposed for this purpose [11, 18, 28], just to name a few. These methods can be classified into two categories, feature-based and window-based methods. However, when f_i^h are scored independently, these approaches are equal [17], *i.e.*, $M(h) = \sum M(h_i)$ where $h_i = f_i^h$. This independence assumption is clearly a simplification, but it is essential for efficient detection. In this work, we use the feature-based representation under this assumption.

3.1. Multi-class Hough Transform

Let us consider a natural multi-class extension of the Hough transform with a shared codebook for all classes in which class label is an additional dimension (similar to separate voting in [24]). Training in this scheme is composed of two stages. Firstly, F^{train} is clustered into a codebook K. In the second stage, a score function $s_k(l, c)$ is learned for each codebook entry using the location and class labels of the training data. For detection, appearance of every feature f_i^h in a hypothesis is assigned to a codebook entry k and a weight is assigned to it using the scoring function. In the Hough transform, instead of creating all the possible hypothesis and obtaining their scores, a set of weighted votes with inverse locations are casted in a voting space. Every element of this voting space is a hypothesis and its extent is obtained by backprojection.

Note that representing appearances with a codebook is not limiting us and we can cast other detection schemes under independent assumption into a Hough transform. For instance, consider the detection with a sliding window approach and using HoG [4] features (e.g., [9]). In this case, the codebook is simply composed of the quantized gradient orientations. The score function learns the weight of each orientation occurring at a certain discrete location (HoG cell). If the detection proceeds using Hough-transform, it would result in the same solution as the sliding window.

3.2. Computational Complexity

For discussing the complexity in terms of number of classes $|C|$, we assume that the number of features per test image is $O(|I|)$ and the cost of extracting a single feature is constant. Furthermore, we assume that the number of training images per class, N, is equal for all classes and the number of features per training example is constant. The total number of features $|F^{train}|$ stored in K is then $O(|C| \times |N|)$. If the cost of matching a feature to a codebook entry is $O(|K|)$, as it is in [18, 28], then the cost of detection is

$$\mathcal{L} = O\left(|I| \times \frac{|K|}{|K|} \times |F^{train}|\right) = O(|I| \times |C| \times |N|). \quad (1)$$

assuming that the features are evenly distributed among codebook entries. As can be seen, the complexity of detection does not depend on $|K|$. Note that this does not imply the equality of different codebooks, however. Another important factor when assessing the performance is the accuracy which indeed depends on how the codebook is trained and its size. For example, when sharing features among categories, [28, 13] have shown that the number of codebook entries is growing only sublinearly in $|C|$ and since sharing makes better use of the training data, it increases the accuracy. Another advantage of sharing is that usually the cost of matching a feature is much higher than the cost of voting and in that respect sharing reduces the computational time although $\mathcal{L}$ still grows with $O(|C|)$. In order to scale the detection, one can either decrease the complexity of matching, use less training images, or reduce the number of voting elements which currently is $O(|C| \times |N|)$.

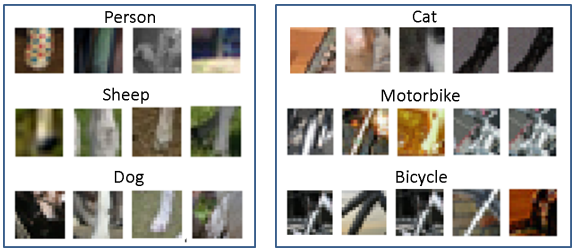
Several approaches [14, 2] proposed matching schemes with cost of $O(log(|K|))$. In this scenario, $\mathcal{L}$ is

$$\mathcal{L} = O\left(|I| \times \frac{log(|K|)}{|K|} \times |C| \times |N|\right). \quad (2)$$

If $|K| = k|C|$, where k does not depend on $|C|$, $\mathcal{L}$ scales with $log(|C|)$ which is ideal. However, our experiments confirm the observation of [28, 13, 29] that in order to afford generalization and handle intra-class variability, the features cannot be very specific to a certain class as implied by $|K| = k|C|$. Hence, the size of the codebook scales sublinearly in $|C|$. And if $|K| = k\, log(|C|)$, we get

$$\mathcal{L} = O\left(|I| \times \frac{log\, log(|C|)}{log(|C|)} \times |C| \times |N|\right), \quad (3)$$

and $O(|C| \times \frac{log\, log(|C|)}{log(|C|)})$ does not scale well in $|C|$.



(a) (b)

Figure 2. Patches clustered in two sample leaves of our multi-class detector trained for VOC'06 dataset. Although the appearance of our middle complexity patches is shared among some classes, it is yet discriminative to other classes. For instance, when a patch is assigned to leaf (a), it is difficult to determine if it belongs to a dog or a person but we can say that it is not coming from a car or a motorbike. This property of middle-complexity features enable building of part-based category hierarchies introduced in Sec. 4.3.

This means that one needs to find a good trade-off. To this end, we propose an approach in which features are shared only between similar classes (Sec. 4.1). Furthermore, the number of voting elements can be reduced using a taxonomy of classes that is automatically obtained from the trained codebook (Sec. 4.3).

4. Proposed Multi-Class Detector

4.1. Training the Shared Multi-class Codebook

For training the codebook, we use a multi-class extension of the class-specific Hough forests [14]. Although we are not limited to this choice, this framework provides us with the flexibility to analyze several issues regarding sharing and multi-class detection.

Class-specific Hough forests discriminatively learn the codebook K and the scoring functions $s_k(l,c)$ from the training data F^{train}. To this end, starting from the root with all features, a binary test is assigned recursively to each node n of the trees that optimally splits features of that node F^n into two sets F^n_{left} and F^n_{right} according to certain criteria. The optimal binary tests are selected as to minimize either location or class uncertainties. We can define class and location uncertainties for multi-class as a natural extension of the criteria introduced in [14] for single class as

$$U^c(F) = |F| \sum_{i=1}^{|C|} -p_{c_i} \cdot log(p_{c_i}) \quad (4)$$

$$U^l(F) = \sum_{j,c_j \neq c_{bg}}^{|F|} \|l_j - \bar{l}\|_2^2 \quad (5)$$

where p_{c_i} is the probability of class c_i and $\bar{l} = \frac{1}{|F|}\sum_j l_j$ is the mean vector of locations.

While (4) enforces the features to be discriminative between objects and background, (5) favors features sharing their location. Optimizing the class and location uncertainties indeed prefers features at each leaf to satisfy both properties. It is important to note, however, that the second criterion (5) is not forcing similar locations as in [28] and it only tries to minimize location uncertainty as much as possible. Nevertheless, this requirement can be restrictive as mentioned earlier (see Fig. 3). Hence, we relax the second criteria to be class dependent letting features to have different locations for different classes:

$$U^l(F) = \sum_{j,c_j \neq c_{bg}}^{|F|} \|l_j - \bar{l}_{c_j}\|_2^2 \quad (6)$$

where $\bar{l}_c = \sum_{j,c_j=c} l_j$.

The class uncertainty measure in (4) treats the background class like any other class and does not ensure well separation of the background from other classes (see Fig. 3(b)). We thus propose a special treatment of the background class by having a cost function measuring the information gain for separating background from any other class given by

$$U^{bg}(F) = |F|(-p_o \cdot log(p_o) - p_{bg} \cdot log(p_{bg})) \quad (7)$$

and another cost function measuring the information gain for separating object classes from each other

$$U^o(F) = |F| \sum_{i=1,c_i \neq c_{bg}}^{|C|} -p_{c_i} \cdot log(p_{c_i}). \quad (8)$$

These two cost functions are then mixed to form the final cost function

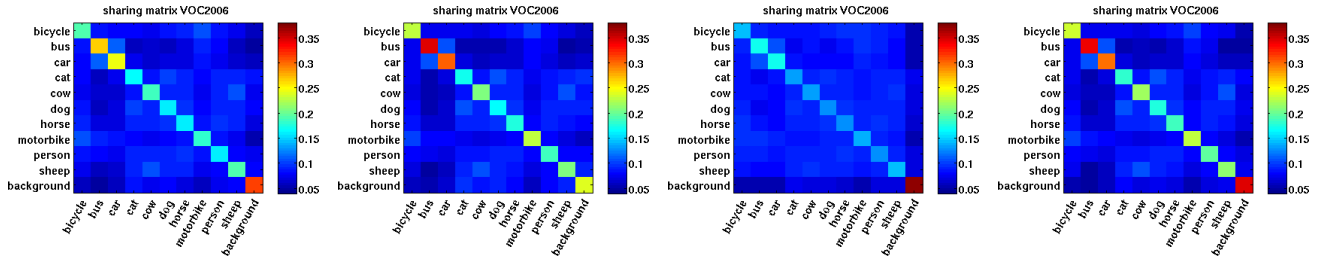
$$U^c(F) = U^o(F) + \lambda U^{bg}(F) \quad (9)$$

and the mixing coefficient λ in our experiments is set to $|C|$. Figure 3 shows sharing matrices of codebooks trained with different criteria.

4.2. Measuring Sharing

For each entry k of the codebook of local appearances K, let us assume that we have pre-calculated a look-up table $Z_k = \{z^k_{c,l}\}$ storing the non-zero weights obtained from the scoring function for every pair of (l,c). Hence, we can redefine the codebook $K = \{a_k, Z_k\}$. For every codebook, we can obtain a sharing matrix $S : |C| \times |C|$, which indicates the degree of sharing between different classes. The sharing matrix can be calculated by considering only appearance, or appearance and location. An element $S(c_i, c_j)$ of a sharing matrix for appearance is calculated by

$$S(c_i,c_j) = \frac{1}{\zeta} \sum_{k=1}^{|K|} |z^k_{c_i}| \sum_{h=1}^{|z^k_{c_j}|} z^k_{c_j,l_h}. \quad \zeta = \sum_{t=1}^{|C|} S(c_i,c_t). \quad (10)$$



(a) (b) (c) (d)

Figure 3. In this figure, the effect of training with various optimization criteria is illustrated. (a) When training according to (4) and (5). (b) By relaxing the location with (6), more distinction is appearing between dissimilar classes but the separation of background class is problematic. (c) If we only separate the foreground and background using (7), the background gets well separated but the classes remain mixed. (d) This is the sharing matrix obtained from (9) for which the classes and background are both well separated. As can be seen, the classes are separated similar to (b) and the foreground and background similar to (c). (best viewed in color)

Similarly, we can calculate a sharing matrix of sharing both appearance and location by

$$S(c_i, c_j) = \frac{1}{\zeta} \sum_{k=1}^{|K|} \sum_{g=1}^{|z^k_{c_i}|} \sum_{h=1}^{|z^k_{c_j}|} z^k_{c_j,l_h} \theta(l_g, l_h). \quad (11)$$

where ζ is again a normalization factor and $\theta(l_g, l_h)$ is a threshold function which is one if $\|l_g - l_h\|_2$ is smaller than a threshold (set to 10 pixels in our implementation). Hence, two features of different classes are considered shared only when they are clustered into a same leaf and if their location is similar.

4.3. Building the Taxonomy

We automatically build a taxonomy of classes by clustering the appearance sharing matrix defined in (10). The appearance sharing matrix is asymmetric and its elements are affinities. For clustering, we transform it into a symmetric dissimilarity matrix D by

$$D = 1 - \frac{1}{2}(S + S^T). \quad (12)$$

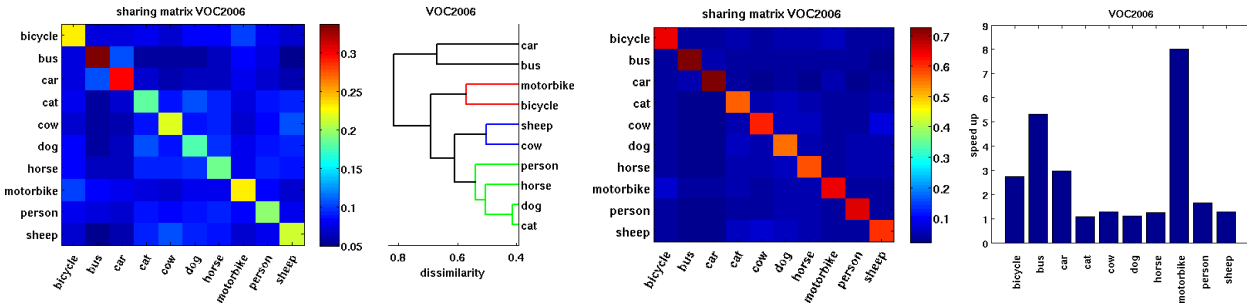
The taxonomy T is obtained by clustering D using the complete-linkage agglomerative clustering. Sharing matrices using (10) and (11) and their corresponding taxonomies are shown in Fig. 4.

In contrast to [16], the hierarchy is derived from the sharing matrix and not from the confusion matrix. This is more efficient since the similarity can be directly computed from the trained codebook without an additional expensive validation procedure that is needed for the confusion matrix. Furthermore, taking the confusion matrix means that the location is also used whereas we would like to have sharing only based on the appearance. Although it is our objective to automatically obtain a taxonomy from the sharing matrix, it is still possible to use WordNet [8]-style hand-crafted semantic taxonomies. In principle, semantic taxonomies enable sharing labels of the learned categories to unseen categories [10]. This transfer is beyond the scope of this paper and we leave it for future work. However, as we will see in the experimental section, the hierarchies that we obtain by clustering the sharing matrix are indeed semantically meaningful; therefore, we do not expect to have a performance drop by using semantic hierarchies.

4.4. Detection with the Taxonomy

Our multi-class detection scheme is a straightforward extension of separate voting introduced in [24] for multi-view. In this scheme, dense features are sampled from the image. Every feature is matched to a codebook entry k and a set of weighted votes are cast to a 4D voting space according to occurrences stored in Z_k. As described in Sec. 3.2, the cost of detection of this method scales with the number of cast votes to the voting space.

In order to reduce this number, we use a taxonomy that has been automatically obtained from the sharing matrix; see Fig. 4(a) for an example taxonomy. The object classes are the leafs of this taxonomy and every internal node t has a subset of classes C_t. We use the taxonomy to efficiently retain a small subset of similar categories for each feature and only vote with those classes. For this purpose, we only look at the appearance of the features. At training time, for every node t and each codebook entry k, we pre-calculate a weight $w^k_t = \sum_{c_j \in C_t} z^k_{c_j}$ and normalize it by the weight at the root and store it in the codebook. At the detection time, a dense set of features are extracted from an image and are assigned to a codebook entry. For every feature f, the taxonomy tree is traversed using breath first search and all the leafs with weights greater than a threshold are retained and the voting is done using those leafs only. Since the weight at each node is the sum of the weights at its children, retaining the leafs with weights bigger than the threshold can be done efficiently. The threshold in our work is node specific and is set to $\alpha \times \frac{linkage}{|C|}$ where $linkage$ is the linkage cost of the parent node (set to one at the root) and α is set to 1.5 in our implementation. In order to increase robustness, we obtain the weights $\hat{w}$ at each node by averaging weights of all the



(a) (b) (c)

Figure 4. Sharing matrices for VOC'06 dataset using our proposed multi-class criteria in Eqs. (6) and (9) for training. (a) Sharing matrix of feature appearances using (10) and its corresponding taxonomy which is automatically obtained by clustering the sharing matrix. When calculating sharing of both appearance and location with (11), the sharing matrix in (b) is obtained. When discriminativity in appearance is combined with location, our multi-class training leads to a very good separation of classes. (c) Speed-up achieved using the taxonomy in (a). The more discriminative features lead to more speed-up. (best viewed in color)

features in the neighborhood of f. In our experiments, we used a neighborhood of 8×8 pixels. The rest of the detection is done in a standard way and the bounding box of each hypothesis is obtained from the backprojection [24].

4.5. Verification

The above mentioned detection algorithm has a very good recall, but this usually comes at the cost of low precision. The main underlying reason for this is the independent assumption which is too crude for classification but vital for fast detection. In addition, the detection system looks only at the features which are well localizable. For example, seeing an elephant's skin tells little about the elephant's center, however, it is useful for classification. To this end, we re-score the hypothesis obtained from the previous step using a more sophisticated classifier (we used [9] for which the source code is kindly provided by the authors). Unlike other methods, *e.g.*, [19], which exhaustively searches a set of bounding boxes in the neighborhood of a detection, the re-scoring in our system is performed only for a single bounding box and a single class, which makes it scalable.

5. Experiments

For our experiments, we use the popular PASCAL VOC 2006 and 2007 datasets [7]. We also consider two baselines for the performance analysis. The first baseline is detection with one-vs-the-rest multi-view Hough forests [24] using joint-voting and then followed by the verification step using [9] as classifiers. The second baseline is the output of the classifiers [9] using sliding window for detection.

The multi-class detector presented in Sec. 4 is single-scale. In order to carry out detection on the multi-scale VOC datasets, the detection is done at multiple scales by rescaling images. We have used the scales: $\sqrt{2}^i$ where $i \in \{-3, -2, \ldots, 8\}$. The range of scales appearing in VOC datasets are different for different classes. For efficiency reasons, a range of scales from the a-forementioned range is selected for each class using the validation set.

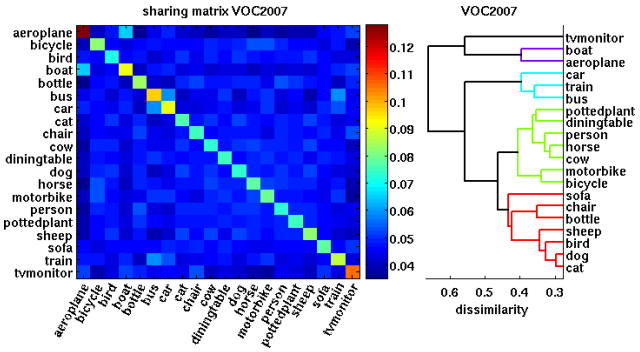


Figure 5. The sharing matrix of appearance and its automatically built taxonomy for the VOC'07 dataset. (best viewed in color)

5.1. Training

Prior to training of the forest, the training objects are cropped from the bounding box annotated images and rescaled to have approximately the same size. A small region around the bounding boxes with 10% of the original bounding box width is added to the cropped image. The rescaling is carried out to have the maximum of width and height not less than 100 pixels and the minimum of them not smaller than 50 pixels. The coordinates at the center of the bounding box of each object is considered as its center. For training each tree, 200 objects per class are randomly selected (when available) where 250 patches are extracted from each. 250 patches are extracted from non-object regions of 200 randomly selected images and used as background patches. The patch size is set to 16×16 pixels which can be considered of intermediate complexity (see Fig. 2). Same image features as [14] are used. A class label is assigned to each feature and the additional view annotations are ignored. Each tree is trained until the depth 20 for VOC'06 and 25 for VOC'07 is reached, or until less than 10 patches for one class are left. The effect of different optimization functions for training is shown in Fig. 3.

Method	bic.	bus	car	cat	cow	dog	hrs.	m.bi.	pers.	shp.	avg
OvA	.16	.13	.07	.04	.18	.03	.15	.16	.11	.12	.114
MC	.37	.12	.11	.02	.14	.05	.08	.21	.05	.12	.127
MC+T.	.38	.13	.12	.05	.15	.03	.11	.12	.05	.12	.132
[9]	.64	.62	.634	.23	.46	.14	.45	.61	.38	.45	.459
OvA+vrf.	.67	.62	.62	.23	.46	.14	.46	.62	.35	.43	.461
MC+vrf.	.68	.64	.65	.20	.47	.14	.44	.64	.38	.43	.465
MC+T.+vrf.	.66	.64	.66	.22	.47	.14	.44	.64	.36	.42	.463

Table 1. Performance comparison of our multi-class method (MC) with the baselines in average-precision for VOC'06 dataset. The first block shows the detection without verification and without non-maxima suppression. MC outperforms one-vs-the-rest (OvA). The taxonomy not only reduces the amount of voting (Fig. 7), it also gives a slight improvement. In the second block, verification is performed with [9]. By using a two-stage method, we are not loosing accuracy compared to [9]. The number of performed verifications is given in Table 2.

5.2. Scalability

Table 6 compares the complexity of our approach to our baselines and the state-of-the-art. Since we are using random decision forests for matching a patch to the codebook, our matching cost is logarithmic in the number of codebook entries. However, as discussed in Sec. 3.2, the complexity of detection also depends on the size of the codebook and the total number of cast votes. Figure 7 compares the codebook size and the number of votes for one-vs-the-rest detectors and our multi-class approach with and without using taxonomies. For this experiment, different number of classes are chosen at random from the VOC'06 dataset and a multi-class detector, with the same settings as above, is trained for them. Although the number of votes scales sublinearly in the multi-class approach, using taxonomies adds further reductions. Figure 4(c) shows the per class speedups when using the taxonomy shown in Fig. 4(a).

5.3. Detection

For detection at every scale, a dense set of features is extracted and matched to the forest and their votes are cast to a voting space similar to [14]. When using the taxonomy, the votes are cast only for the categories with weights bigger than the threshold as described in Sec. 4.4. The voting space is implemented by having a separate accumulator for every class, but since the number of cast votes to this accumulator scales sublinearly, so does the detection. All hypotheses up to a certain threshold for each class are detected and their bounding boxes are estimated using backprojection similar to [24] and no further non-max-suppression is performed before verification. For verification, the final verification classifier [9] is applied to every hypothesis. For fairness in comparison with [9], the same bounding boxes and non-maxima suppression are used after verification. Tables 1 and 3 summarize the accuracy of our two-stage multi-class detector. Table 2 compares the number of windows passed to the verification for both VOC'06 and VOC'07 datasets.

Method	#windows	#verifications
Our approach-VOC'06 (10 cat.)	1321	1321
Our approach-VOC'07 (20 cat.)	1778	1778
[9]-VOC'07(20 cat.)	42278	833141

Table 2. Our multi-class detection approach reduces the number of windows for verification per image by three orders of magnitude. Unlike sliding window approaches, our method assigns a class to each returned window. This eases the verification as one classifier should be evaluated per window, and this without compromising accuracy; see Tables 1 and 3.

Since in our system the number of evaluated windows is three orders of magnitude lower than a sliding window approach, there is the possibility of using more sophisticated classifiers with many features [15]. The actual run time (seconds per image) of our method for single class is 35sec, but only 100sec when detection all 20 classes jointly with the taxonomy. Comparing these numbers with the very fast verification detector in [9], with 7sec per image for single class and 134sec for 20 classes, there is already a benefit for less than 20 classes. Unlike our method, the generic object detector in [1] compromises accuracy for the sake of speed.

6. Conclusions

In this paper, we have presented a scalable multi-class detector by treating location and appearance of features differently. The detection complexity of our method is sublinear in the number of classes. Our approach also benefits from sharing features and an automatically built category taxonomy for robust scalability without compromising accuracy. We have further shown how the detection scheme can be combined in a scalable manner with a verification classifier. Yet, in the current implementation only simple intensity and HoG features have been used which are not appropriate for textured classes like cats and dogs. In addition, although high resolution training data is provided, the training completely discards this by rescaling all training images to relatively small sizes. In the future, we are planning to overcome these shortcomings by using texture features and building a multi-resolution representation of patches similar to [23]. Further, we plan to run our method on ImageNET [5] with many classes.

Acknowledgments: This work has been funded by the Swiss National Fund (SNF) project CASTOR (200021-118106) and the EU projects DIRAC, IURO and RADHAR.

References

[1] B. Alexe, T. Deselaers, and V. Ferrari. What is an object? In *CVPR*, 2010.

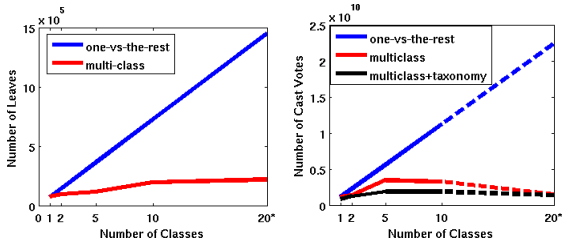
[2] O. Boiman, E. Shechtman, and M. Irani. In defense of nearest neighbor based image classification. In *CVPR*, 2008.

[3] J. Carreira and C. Sminchisescu. Constrained Parametric Min-Cuts for Automatic Object Segmentation. In *CVPR*, 2010.

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

Methods	|K|	Match. in |K|	No. of Votes	Training
Feature-based methods				
ISM [18]	linear	linear	linear	quadratic
Hierarchy of parts[13]	sublinear	linear	linear	linear
Coarse-to-fine[12]	sublinear	linear	sublinear	linear
HoughForest [14]	linear	sublinear	linear	linear
This paper	**sublinear**	**sublinear**	**sublinear**	**linear**
Sliding Window Methods				
Joint-boost [28]	sublinear	linear	linear	quadratic
Part-model [9]	constant	constant	linear	linear

Figure 6. Complexity of different methods for object detection in the number of classes. For [13, 12], the constellation model of contour features is used. Although both our method and [12] are sublinear, [12] uses coarse-to-fine representation of contour features at fixed locations which leads to accuracy loss.



(a) (b)

Figure 7. Our scalable method shows a sublinear growth in the size of codebook (a) and in the number of votes (b) as opposed to linear in one-vs-the-rest. The taxonomy further reduces the number of votes. (20* uses VOC'07)

Method	aerop.	bicyc.	bird	boat	bottle	bus	car	cat	chair	cow	dinin.	dog	horse	motor.	pers.	potte.	sheep	sofa	train	tvmo.	avg.
MC	.01	.13	.09	.01	.02	.01	.08	.10	.01	.10	.01	.02	.11	.14	.05	.03	.10	.01	.10	.11	.060
MC+T.	.01	.15	.09	.01	.02	.01	.07	.10	.01	.05	.01	.02	.12	.13	.03	.02	.10	.02	.10	.10	.057
[9]	.28	.53	.10	.15	.24	.46	.49	.16	.20	.24	.19	.11	.54	.45	.37	.12	.17	.30	.42	.40	.294
MC+verif.	.26	.56	.10	.11	.21	.47	.50	.16	.19	.23	.20	.12	.51	.45	.37	.12	.17	.29	.41	.38	.292
MC+T.+verif.	.25	.56	.10	.10	.21	.48	.51	.16	.19	.24	.20	.11	.52	.45	.37	.12	.17	.29	.41	.38	.291

Table 3. Performance evaluation on VOC'07 dataset in average precision (AP). Our scalable method with verification (MC+verif.) achieves very similar results to the baseline [9].

[5] 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.

[6] I. Endres and D. Hoiem. Category independent object proposals. In *ECCV*, 2010.

[7] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88:303–338, 2010.

[8] C. Fellbaum. *WordNet An Electronic Lexical Database*. The MIT Press, Cambridge, MA. London, May 1998.

[9] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *TPAMI*, 32(9):1627 – 1645, 2009.

[10] R. Fergus, H. Bernal, Y. Weiss, and A. Torralba. Semantic label sharing for learning with many categories. In *ECCV*, 2010.

[11] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *CVPR*, 2003.

[12] S. Fidler, M. Boben, and A. Leonardis. A coarse-to-fine taxonomy of constellations for fast multi-class object detection. In *ECCV*, 2010.

[13] S. Fidler and A. Leonardis. Towards scalable representation of object categories: Learning a hierarchy of parts. In *CVPR*, 2007.

[14] J. Gall and V. Lempitsky. Class-specific hough forests for object detection. In *CVPR*, 2009.

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

[16] G. Griffin and P. Perona. Learning and using taxonomies for fast visual categorization. In *CVPR*, 2008.

[17] A. Lehmann, B. Leibe, and L. Van Gool. Fast prism: Branch and bound hough transform for object class detection. *IJCV*, pages 1–23, 2010.

[18] B. Leibe, A. Leonardis, and B. Schiele. Robust object detection with interleaved categorization and segmentation. *IJCV*, 77(1-3):259–289, 2008.

[19] S. Maji and J. Malik. Object detection using a max-margin hough transform. In *CVPR*, 2009.

[20] M. Marszalek and C. Schmid. Constructing category hierarchies for visual recognition. In *ECCV*, 2008.

[21] M. McCloskey and E. Palmer. Visual representation of object location: Insights from localization impairments. *Curr. Direc. in Psych. Scie.*, pages 5–25, 1996.

[22] B. Ommer and J. Malik. Multi-scale object detection by clustering lines. In *ICCV*, 2009.

[23] D. Park, D. Ramanan, and C. Fowlkes. Multiresolution models for object detection. In *ECCV*, 2010.

[24] N. Razavi, J. Gall, and L. Van Gool. Backprojection revisited: Scalable multi-view object detection and similarity metrics for detections. In *ECCV*, 2010.

[25] J. G. Rueckl, K. R. Cave, and S. M. Kosslyn. Why are 'what' and 'where' processed by separate cortical visual systems? a computational investigation. *J. Cognitive Neuroscience*, 1(2):171–186, 1989.

[26] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *CVPR*, 2008.

[27] R. Stefan, V. Athitsos, Q. Yuan, and S. Sclaroff. Reducing jointboost-based multiclass classification to proximity search. In *CVPR*, 2009.

[28] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing visual features for multiclass and multiview object detection. *TPAMI*, 29(5):854–869, 2007.

[29] S. Ullman, M. Vidal-Naquet, and E. Sali. Visual features of intermediate complexity and their use in classification. *Nature neuroscience*, (5):682–687, 2002.

[30] L. G. Ungerleider and J. V. Haxby. 'what' and 'where' in the human brain. *Curr. Opin. in Neurobio.*, 4:157–165, 1994.

[31] Y. Zhang and T. Chen. Implicit shape kernel for discriminative learning of the hough transform detector. In *BMVC*, 2010.