Incremental learning of object detectors using a visual shape alphabet

Andreas Opelt, Axel Pinz
Inst. of Electrical Measurement and Sign. Proc.
Graz University of Technology, Austria

Andrew Zisserman
Dept. of Engineering Science
University of Oxford, UK

Abstract

We address the problem of multiclass object detection. Our aims are to enable models for new categories to benefit from the detectors built previously for other categories, and for the complexity of the multiclass system to grow sublinearly with the number of categories. To this end we introduce a visual alphabet representation which can be learnt incrementally, and explicitly shares boundary fragments (contours) and spatial configurations (relation to centroid) across object categories.

We develop a learning algorithm with the following novel contributions: (i) AdaBoost is adapted to learn jointly, based on shape features; (ii) a new learning schedule enables incremental additions of new categories; and (iii) the algorithm learns to detect objects (instead of categorizing images). Furthermore, we show that category similarities can be predicted from the alphabet.

We obtain excellent experimental results on a variety of complex categories over several visual aspects. We show that the sharing of shape features not only reduces the number of features required per category, but also often improves recognition performance, as compared to individual detectors which are trained on a per-class basis.

1 Introduction

Many recent papers on object category recognition have proposed models and learning methods where a new model is learnt individually and independently for each object category [1, 5, 10, 12]. In this paper we investigate how models for multiple object categories, or for multiple visual aspects of a single category, can be built incrementally so that new models benefit from those created earlier. Such models and methods are necessary if we are to achieve the long sought after system that can recognize tens of thousands of categories: we do not want to be in a position where, in order to add one more category (after number 10,000), we have to retrain everything from scratch. Of course, the constraint is that our recognition performance should at least equal that of methods which learn object category models individually.

In this paper we concentrate on object models consisting of an assembly of curve fragments. This choice is because the curve fragments more closely represent the object shape (than the more commonly used appearance patches [3, 10, 12]). This representation can be complemented by adding appearance patches, though we do not investigate that here. Our object model is similar to those of [13, 14, 15] and is briefly reviewed in section 2.

We introduce a novel joint learning algorithm which is a variation on that of Torralba *et al.* [16], where weak classifiers are shared between classes. The principal differences are that our algorithm allows *incremental* as well as joint learning, and we can control the degree of sharing. Less significant differences follow from the use of the boundary fragment model [13] in that we learn an object *detector* (rather than the *classification* of an image window, and detection by scanning over the whole image as is done in [16]). The main benefits of the approach, over individual learning of category detectors, are: (i) that we need less training data when sharing across categories; and (ii) that we are able to add new categories incrementally making use of already acquired knowledge.

Others have also used information from previously learnt classes. For example, Fei-Fei *et al.* [4] used prior information from previously learnt categories to train a generative probabilistic model for a novel class, and Bart and Ullman [2] introduced a cross-generalization method where useful patches for one category guide the search within the pool of possible patches for a new, but similar, category. Krempp *et al.* [9] have a similar objective of incremental learning of categories and a shared alphabet. However, their category model and learning algorithm differ substantially from that proposed here.

A brief outline of the paper is as follows: we start with an introduction of the BFM and show that we need to train only a few relevant aspects per category. Next, we present the incremental learning of the visual alphabet, which is shared over categories. Similarly, our detectors are learnt incrementally and can be shared. Finally, our experiments show that this sharing leads to a sublinear growth of required alphabet entries / detectors, but maintains excellent detection performance.

2 The boundary fragment model (BFM)

We present a very brief overview of our previous work which introduced a boundary fragment model (BFM) detector (see [13] for details). The BFM consists of a set of curve fragments representing the edges of the object, both internal and external (silhouette), with additional geometric information about the object centroid (in the manner of [10]). A BFM is learnt in two stages. First, random boundary fragments γ_i are extracted from the training images. Then costs $K(\gamma_i)$ are calculated for each fragment on a validation set. Low costs are achieved for boundary fragments that match well on the positive validation images, not so well on the negative ones, and have good centroid predictions on the positive validation images. Second, combinations of $k = 2$ boundary fragments are learnt as weak detectors (not just classifiers) within an AdaBoost [6] framework. Detecting instances of the object category in a new test image is done by applying the weak detectors and collecting their votes in a Hough voting space. An object is detected if a mode (obtained using Mean-Shift mode estimation) is above a detection threshold. Following the detection, boundary fragments that contributed to that mode are backprojected into the test image and provide an object segmentation. An overview of the detection method is shown in figure 1.

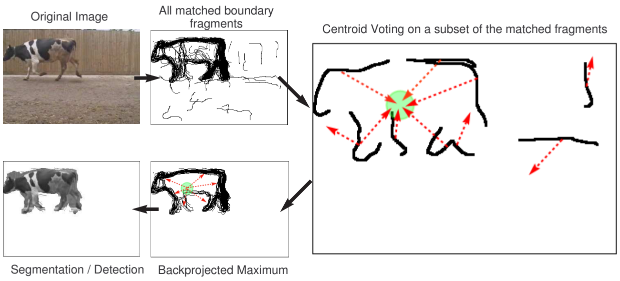

Original Image
All matched boundary fragments
Centroid Voting on a subset of the matched fragments
Segmentation / Detection
Backprojected Maximum

Figure 1: Overview of object detection with the boundary fragment model (BFM).

3 On multiple aspects

We want to enable an object to be detected over several visual aspects. The BFM implicitly couples fragments via the centroid, and so is not as flexible as, say, a “bag of” features model where feature position is not constrained. In this section we investigate qualitatively the tolerance of the model to viewpoint change. The evaluation is carried out on the ETH-80 dataset. This is a toy dataset (pun intended), but is useful here for illustration because it contains image sets of various instances of categories at controlled viewpoints.

We carry out the following experiment: a BFM is learnt from instances of the cow category in side views. The model is then used to detect cows in test images which vary in two ways: (i) they contain cows (seven *different* object instances) over varying viewpoints – object rotation about a vertical and horizontal axis (see figure 2); (ii) they contain instances of other categories (horses, apples, cars . . .), again over varying viewpoints.

Figure 2 shows the resulting Hough votes on the centroid, averaged over the seven cow instances for a number of rotations. It can be seen that the BFM is robust to significant viewpoint changes with the mode still clearly defined (though elongated). The graph in figure 3 summarizes the change in the detection response averaged over the different cows or other objects under rotation about a vertical axis (as in the top row of figure 2). Note that the cow detection response is above that of other non-cow category objects. The side-trained BFM can still discriminate the object class based on detection responses with rotations up to 45 degrees in both directions. In summary: the BFM trained on one visual aspect can correctly detect the object class over a wide range of viewpoints, with little confusion with other object classes. Similar results are obtained for BFM detectors learnt for other object categories (e.g. horses), whilst for some categories with greater invariance to viewpoint (e.g. bottles) the response is even more stable. These results allow us to cut down the bi-infinite space of different viewpoints to a few category relevant aspects. These aspects allow the object to be categorized and also to predict its viewpoint.

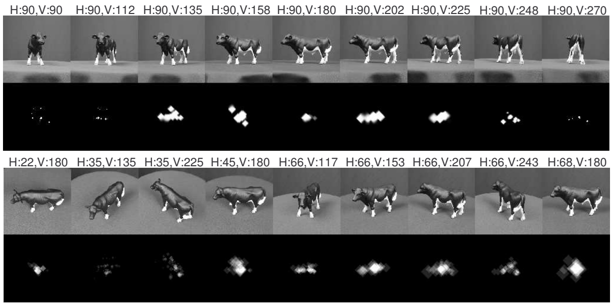


Figure 2: Robustness of the BFM to viewpoint changes under rotations about a vertical (V) or horizontal (H) axis. Top row: rotations about a vertical axis. Bottom row: rotations about both vertical and horizontal. The viewpoint angles are given above each image.

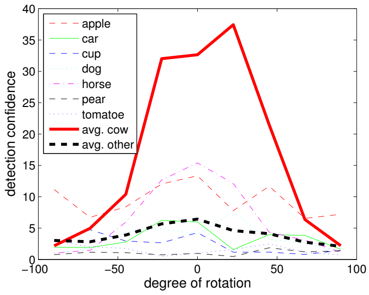


Figure 3: The detection response of a BFM trained on cows-side, and tested on cows rotated about a vertical axis and on other objects.

4 Learning the shape based alphabet incrementally

In this section we describe how the basic alphabet is assembled for a set of classes. Each entry in the alphabet consists of three elements: (i) a curve fragment, (ii) associated vectors specifying the object's centroid, and (iii) the set of categories to which the vectors apply. The alphabet can be enlarged in two ways: (i) adding additional curve fragments, or (ii) adding additional vectors to existing curve fragments – so that a fragment can vote for additional object's centroids. Pairs of curve fragments are used to construct the weak detectors of section 5.

We start from a set of boundary fragments for each category. This set is obtained from the fragment extraction stage (see section 2 or [13]) by choosing fragments whose costs on the validation set of the category are below a given threshold th_K. Typically this threshold is chosen so that there are about 100 fragments available per category. Our aim is to learn a common alphabet from these pooled individual sets that is suitable for all the categories one wants to learn.

4.1 Building the alphabet and sharing of boundary fragments

In a sequential way each boundary fragment from each category is compared (using Chamfer distance) to all existing alphabet entries. If the distance to a certain alphabet entry is below a similarity threshold th_{sim}, the geometric information (for the centroid vote) is updated. If the existing alphabet entry originates from another category than the boundary fragment we are currently processing, we also update the information for which categories this entry is suitable. This is the first case where boundary fragments are shared. This sharing is just based on the boundary fragment similarity.

But there is more information that can be used for sharing. The second possibility of sharing is achieved by evaluating each boundary fragment on the validation sets of all other categories. This results in average matching costs of the boundary fragment on all these other categories. These costs indicate how suitable the boundary fragment is for each of the other categories. The straight forward way of sharing is now that each alphabet entry whose boundary fragment has costs below th_K on a certain category is also shared for that category. However, costs are low if the boundary fragment matches well on the validation images of that category *and* gives a reliable centroid prediction. The final possibility of sharing is where the boundary fragment matches well, but additional centroid vectors are associated for the fragment for the new category. Figure 4 shows an example of a boundary fragment extracted from one category also matching on images of another class (or aspect). The first column shows the original boundary fragment (in red/bold) on the training image from which it was learnt (green/bold cross showing the true object centroid, and blue/bold the centroid vote of this boundary fragment). The other columns show sharing on another category (first row), and within aspects of the same category (second row). Note, that we share the curve fragment and update the geometric information.

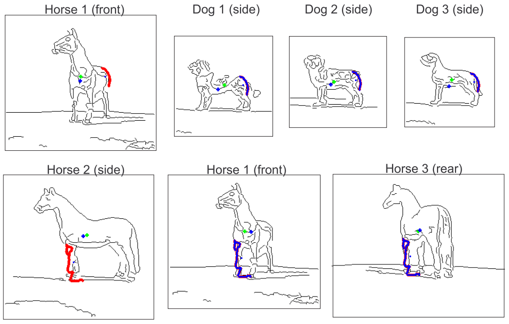


Figure 4: Sharing of boundary fragments over categories (first row) and aspects (second row).

4.2 Class similarities on the alphabet level

We now have alphabet entries for a number of classes. Using this information we can preview class similarities before training the final detector. A class similarity matrix is calculated where each element is a count of the number of alphabet entries in common between the classes. In turn, the classes can be agglomeratively clustered based on their similarity. For this clustering the normalized columns of the similarity matrix provide feature vectors and Euclidean distance is used as a distance measure. An example similarity matrix and dendrogram (representing the clustering) are shown in figures 8(a) and (b) respectively.

5 Incremental Joint-Adaboost Learning

In this section we describe the new Adaboost based algorithm for learning the strong object detectors. It is designed to scale well for many categories and to enable incremental and/or joint learning. It has to do two jobs: (i) select pairs of fragments to form the weak detectors (see section 2); and (ii) select weak detectors to form the strong detector for each object category. Sharing occurs at two levels: first, at the alphabet level where an alphabet entry may be applicable to several categories; second, at the weak detector level, where weak detectors are shared across strong detectors.

The algorithm can operate in two modes: either joint learning (as in [16]); or incremental learning. In both cases our aim is a reduction in the total number of weak detectors required compared to independently learning each class. For C classes this gain can be measured by $\sum_{i=1}^{C} T_{c_i} - T_s$ (as suggested in [16]) where T_{c_i} is the number of weak detectors required for each class trained separately (to achieve a certain error on the validation set) and T_s is the number of weak detectors required when sharing is used. In the separate training case this sum is $O(C)$, whereas in the sharing case it should grow sub-linearly with the number of classes. The algorithm optimizes an error rate E_n over all classes.

Joint learning: involves for each iteration searching for the weak detector for a subset $S_n \in \mathcal{C}$ that has the lowest accumulated error E_n on all classes $\mathcal{C}$. Subsets might be e.g. $S_1 = \{c_2\}$ or $S_3 = \{c_1, c_2, c_4\}$. A weak detector only fits for a category if ϵ^{c_i} on this category c_i is below 0.5 (and is rejected otherwise). E_n is the sum of all class specific errors ϵ^{c_i} if $c_i \in S_n$ and a penalty error ϵ_p (0.6 in our implementation) otherwise. Searching for a minimum of E_n over a set of subsets S_n guides the learning towards sharing weak detectors over several categories. We give a brief example of that behavior: imagine we learn three categories, c_1, c_2 and c_3. There is one weak detector with $\epsilon^{c_1} = 0.1$ but this weak detector does not fit any other category ($\epsilon^{c_2} > 0.5$ and $\epsilon^{c_3} > 0.5$). Another weak detector can be found with $\epsilon^{c_1} = 0.2$, $\epsilon^{c_2} = 0.4$ and $\epsilon^{c_3} = 0.4$. In this case the algorithm would select the second weak detector as its accumulated error of $E_n = 1.0$ is smaller than the error of the first weak detector of $E_n = 1.3$ (note that for each category not shared ϵ_p is added). This makes the measure E_n useful to find detectors that are suitable for both distinguishing a class from the background, *and* for distinguishing a class from other classes. Clearly, the amount of sharing is influenced by the parameter ϵ_p which enables us to control the degree of sharing in this algorithm. Instead of exploring all $2^C - 1$ possible subsets S_n of the jointly trained classes $\mathcal{C}$, we employ the maximally greedy strategy from [16]. This starts with the first class that achieves alone the lowest error on the validation set, and then incrementally adds the next class with the lowest training error. The combination which achieves the best overall detection performance over all classes is then selected. [16] showed that this approximation does not reduce the performance much.

Incremental learning: implements the following idea: suppose our model was jointly trained on a set of categories $\mathcal{C}_L = \{c_1, c_2, c_3\}$. Hence the "knowledge" learnt is contained in a set of three strong detectors $H_L = \{H_1, H_2, H_3\}$ which are composed from a set of weak detectors h_L. The number of these weak detectors depends on the degree of sharing and is defined as $T_s \leq \sum_{i=1}^{C} T_{c_i}$ ($C = 3$ here). Now we want to use this existing information to learn a detector for a new class c_{new} (or classes) incrementally. To achieve this, one can search already learnt weak detectors h_L to see whether they are also suitable ($\epsilon^{c_{new}} < 0.5$) for the new class. If so, these existing weak detectors are also used to form a detector for the new category and only a reduced number of new weak detectors have to be learnt using the joint learning procedure. Note that joint and incremental training reduces to standard Boosting if there is only one category.

Weak detectors: are formed from pairs of fragments. The possible combinations of k fragments define the feature pool (the size of this set is the binomial coefficient of k and the number of alphabet entries). This means for each sharing of each iteration we must search over all these possibilities to find our best weak detector. We can reduce the size of this feature pool by using only combinations of boundary fragments which can be shared over the same categories as candidates for weak detectors. E.g. it does not make much sense to test a weak detector which is combined from a boundary fragment representing a horses leg and one that represents a bicycle wheel if the boundary horses leg never matches in the bike images.

Details of the algorithm: The algorithm is summarized in figure 5. We train on C different classes where each class c_i consists of N_{c_i} validation images, and a set of N_{bg} background validation images (which are shared for all classes and are labeled ℓ_i^0). The total number of validation images for all classes and background is denoted by N. The weights are initialized for each class separately. This results in a weight vector w_i^c of length N for each class c_i, normalized with respect to the varying number of positive validation images N_{c_i}. In each iteration a weak detector for a subset S_n is learnt. To encourage the algorithm to focus also on the categories which were not included in S_n

we vary the weights of these categories slightly for the next iteration ($\epsilon^c = p, \forall c \notin S_n$, with $p = 0.47$ in our implementation).

```
Input: Validation images $(I_1, \ell_1^0), \ldots, (I_N, \ell_N^C)$,
$\ell_i^c \in \{\mathcal{C}, -1\}, N = N_{bg} + \sum_{i=1}^{C} N_{c_i}$.
Initialization: Set the weight matrices $w_i^c$:
$w_i^c = \begin{cases} \frac{1}{2N_{c_i}} & \text{if } \ell_i = c. \\ \frac{1}{2(N_{bg}+\sum_{i=1,c_i \neq \ell_i}^{C} N_{c_i})} & \text{else} \end{cases}$
Learn incrementally:
For $c_i = 1 : C$
  For $h_L(I, S_n) \in H_L(I, c)$
    if $\epsilon^{c_i} < 0.5$: $h_L = h_L(I, S_n \cap c_i)$, update $w_i^c, t = t + 1$
$T_{c_i} = T_{c_i} + 1$
For $t = 1, ..., T_{max}$
  1. For $n = 1, 2, .., \frac{C(C+1)}{2}$
     (a) Find the best weak detector $h_t(I, S_n)$ w.r.t.
         the weights $w_i^{S_n}$.
     (b) Evaluate error:
         $E_n = \begin{cases} \sum_c^C \epsilon^c & \text{if } \epsilon^c < \frac{1}{2}, \forall c \in S_n \\ C & \text{else} \end{cases}$
     with $\epsilon^c = \begin{cases} \frac{\sum_{i=1}^{N} w_i^c \cdot (\frac{1}{2}(\ell_i^c - h_t(I_i, S_n))^2)}{\sum_{i=1}^{N} w_i^c} & \text{if } \ell_i \in S_n, \\ \epsilon_p & \text{otherwise.} \end{cases}$

  2. Get best sharing by selecting: $n = argmin_n E_n$
     and pick corresponding $h_t, S_n$

  3. Update additive model and weights:
     $H(I, c) = H(I, c) + \alpha_t h_t(I, S_n)$
     $w_i^c \leftarrow w_i^c \cdot \alpha^{\ell_i^c h_t(I_i, c)}$
     with $\alpha_t = \frac{1}{2} \log\left(\frac{1-\epsilon^c}{\epsilon^c}\right)$, and $\epsilon^c = p$ for $c \notin S_n$

  4. Update $T_{c_i}$, and if $T_{c_i} \geq T \, \forall c_i \rightarrow STOP$
```

Figure 5: Incremental joint-Adaboost learning algorithm.

6 Experiments

We will measure detector performance in two ways: first, by applying the detector to a category specific test set (positive images vs. background). The measure used is the Recall Precision Curve (RPC)-equal-error rate. This rate is commonly used for detection and pays respect to false positive detections (see [1] for more details); second, by a confusion table computed on a multi-class testset. Note that a detection is correct if $\frac{area(box_{pred} \cap box_{gt})}{area(box_{pred} \cup box_{gt})} \geq 0.5$, with box_{pred} being the predicted bounding box and box_{gt} the bounding box denoting the ground truth.

The detectors are trained in three ways: (i) independently using the category's validation set (images with the object, and background images); (ii) jointly over multiple categories; and (iii) incrementally. We compare performance, learning complexity, and efficiency of the final strong detectors over these three methods.

For all experiments training is over a fixed number of weak detectors $T = 100$ per class (for C classes the maximum number of weak detectors is $T_{max} = T \cdot C$). This means we are not searching and comparing the learning effort for a certain error rate (as is done in [16]), but we report the RPC-equal-error-rate for a certain learning effort (namely T weak detectors). Keeping track of the training error is more difficult in our model, as we detect in the Hough voting space manner of [13] instead of classifying subwindows like [16].

The experiments are organized as follows: First we briefly explain how the detection procedure works for the multi-class case. Then we specify the used data, and show results on the plain alphabet followed by a comparison of incremental and joint learning. Finally we present results of learning many categories independently or jointly.

Detection algorithm: For a test image our task is to detect one or more objects from C classes. This is carried out by the standard detection procedure (see section 2, and for details [13]) extended to the multi-class case. All weak detectors trained for the C classes are applied to the test image. For each class we then manage a separate Hough voting space and add votes for all weak detectors that matched on that image and are shared by that category (included in the strong detector for that category). Finally, we search each of the voting spaces for maxima and detect an object of class c_i if there is a maximum in the corresponding voting space above threshold.

Dataset: we have combined different categories from several available datasets (at [8]) together with new images from Google Image Search, in order to assemble a dataset containing 17 categories of varying complexity and aspect. Figure 6 overviews the dataset, giving an example image for each of the 17 categories. Table 1 summarizes the data used for training, validation and testing.

We use the same test set as [5] for the first four categories so that our performance can be compared to others (although fewer training images are used). The same is done for category 11 (CowSide) so that performance can be compared with [10]. For the other categories we are not directly comparable as subsets of the training and test data have been selected. As background images we used a subset of the background images used in [5] and [12] (the same number of background as positive training images). To determine to what extent the model confuses categories, we select a multiclass test dataset $\mathcal{M}$ which consists of the first 10 test images from each category[1].

The alphabet: Figure 7 shows entries of the alphabet trained on horses only. This nicely illustrates the different properties of each entry: shape and geometric information for the centroid. When we train on 17 categories each of the

[1] The whole dataset is available at [7].

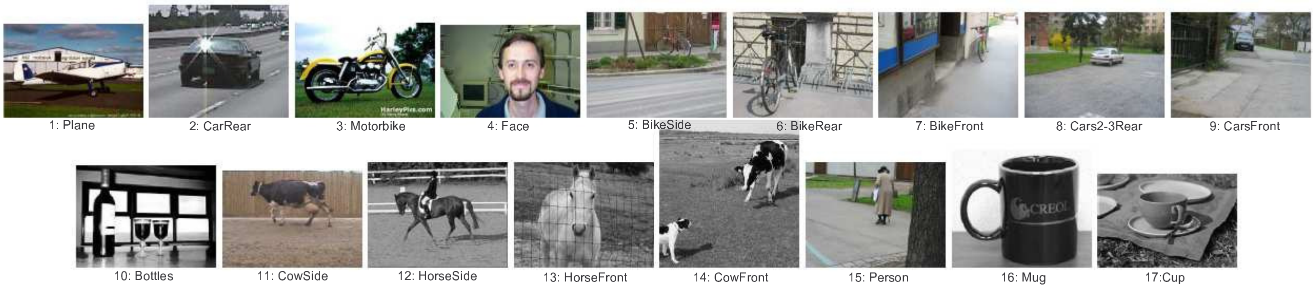


Figure 6: Example images of the 17 different categories (or aspects) used in the experiments.

C	Name	train	val	test	source
1	Plane	50	50	400	Caltech [5]
2	CarRear	50	50	400	Caltech [5]
3	Motorbike	50	50	400	Caltech [5]
4	Face	50	50	217	Caltech [5]
5	BikeSide	45	45	53	Graz02 [12]
6	BikeRear	15	15	16	Graz02 [12]
7	BikeFront	10	10	12	Graz02 [12]
8	Cars2-3Rear	17	17	18	Graz02 [12]
9	CarsFront	20	20	20	Graz02 [12]
10	Bottles	24	30	64	ImgGoogle [13]
11	CowSide	20	25	65	[11]
12	HorseSide	30	25	96	ImgGoogle
13	HorseFront	22	22	23	ImgGoogle
14	CowFront	17	17	17	ImgGoogle
15	Person	19	20	19	Graz02 [12]
16	Mug	15	15	15	ImgGoogle
17	Cup	16	15	16	ImgGoogle

Table 1: The number of training, validation and test images.

alphabet entries is on average shared over approximately 5 categories. The alphabet can be used to take a first glance at class similarities. Figures 8(a) and (b) show the results of the procedure described in section 4.2. The correlations visible in the similarity matrix are due to alphabet entries that can be shared over categories. The dendrogram for the 17 categories shows some intuitive similarities (e.g. for the CarRear and CarFront classes).

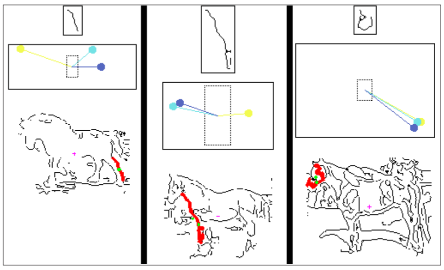
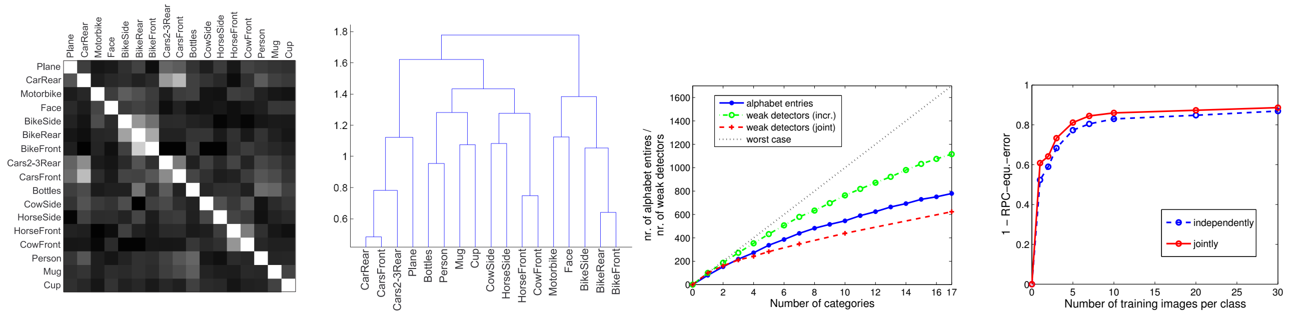


Figure 7: Example alphabet entries from learning only horses. Each column shows the shape of the boundary fragment (top), the associated centroid vector for this entry (middle), and the training image where the boundary fragment (shown in red/bold) was extracted.

Incremental learning: Here we investigate our incremental learning at the alphabet level, and on the number of weak detectors used. We compare its sharing abilities to independent and joint learning. A new category can be learnt incrementally, as soon as one or more categories have already been learnt. This saves the effort of a complete retraining procedure, but only the new category will be able to share weak detectors with previously learnt categories, not the other way round. However, with an increasing number of already learnt categories the pool of learnt weak detectors will enlarge and give a good basis to select shareable weak detectors for the new unfamiliar category. We thus can expect a sublinearly growing number of weak detectors when adding categories incrementally. The more similar categories are already known the more can be shared. This can be confirmed by a simple experiment where the category HorseSide is incrementally learnt, based on the previous knowledge of an already learnt category CowSide, showing that 18 weak detectors are shared. In comparison, the joint learning shares a total of 32 detectors (CowSide also benefits from HorseSide features). For the 17 categories incremental learning shows its advantage at the alphabet level. We observe (see figure 8(c)) that the alphabet requires only 779 entries (worst case approximately 1700 for our choice of the threshold th_K, giving roughly a set of 100 boundary fragments per category).

Figure 8(c) shows the increase in number of shared weak detectors, when a number of new categories are added incrementally, one category at a time. Assuming we do learn 100 weak detectors per category the number of the worst case (1700) can be reduced to 1116 by incremental learning. Learning all categories jointly reduces the number of used weak detectors even further to 623. However, a major advantage of the incremental approach is the significantly reduced computational complexity. While the joint learning with I validation images requires $O(2^C I)$ steps for each weak detector, incremental learning has a complexity of only $O(h_L I)$ for those weak classifiers (from already learnt



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

Figure 8: (a) Similarity matrix of alphabet entries for the different categories (brighter is more similar). (b) Dendrogram generated from this similarity matrix. (c) The increase in the number of alphabet entries and weak detectors when adding new classes incrementally or training a set of classes jointly. The values are compared to the worst case (linear growth, dotted line). For weak detectors the worst case is training independent and given by ($\sum_{i=1}^{C} T_{c_i}$), and for the alphabet we approximate the worst case by assuming an addition of 100 boundary fragments per category. Classes are taken sequentially (Planes(1), CarRear(2), Motorbike(3), ...). Note the sublinear growth. (d) Error averaged for 6 categories (Planes, CarRear, Motorbike, Face, BikeSide and HorseSide) either learnt independently or jointly with a varying number of training images per category.

weak classifiers) that can be shared. One could use the information from the dendrogram from figure 8(b) to find out the optimal order of the classes for the incremental learning, but this is future work.

Joint learning: First we learn detectors for different aspects of cows, namely the categories CowSide and CowFront independently, and then compare this performance with joint learning. For CowSide the RPC-equal-error is 0% for both cases. For CowFront the error is reduced from 18% (independent learning) to 12% (joint learning). At the same time the number of learnt weak hypotheses is reduced from 200 to 171. We have carried out a similar comparison for horses which again shows the same behavior. This is due to the reuse of some information gathered from the side aspect images to detect instances from the front. Information that is shared here are e.g. legs, or parts of the head. This is precisely what the algorithm should achieve – fewer weak detectors with the same or a superior performance. The joint algorithm has the opportunity of selecting and *sharing* a weak detector that can separate both classes from the background. This only has to be done once. On the other hand, the independent learning does not have this opportunity, and so has to find such a weak detector for each class.

In figure 8(d) we show that joint learning can achieve better performance with less training data as a result of sharing information over several categories (we use 6 categories in this specific experiment).

Finally we focus on many categories, and compare independent learning performance to that achieved by learning jointly. Table 2 shows the detection results on the categories test set (category images and background images), denoted by $\mathcal{T}$ and on the multiclass test set ($\mathcal{M}$) for both cases. It also gives comparisons to some other methods that used this data in the single category case where we used the same test data. The joint learning procedure does not significantly reduce the detection error (although we gain more than we loose), but we gain in requiring just 623 weak detectors instead of the straightforward 1700 (i.e. 100 times the number of classes for independent learning). Errors are more often because of false positives than false negatives. We are superior or similar in our performance compared to state-of-the-art approaches (note that classification is easier than detection) as shown in table 2. Looking at the multiclass case (I, $\mathcal{M}$, and J, $\mathcal{M}$, in error per image), we obtain comparable error rates for independent and joint learning. Figure 9 shows examples of weak detectors learnt in this experiment, and their sharing over various categories.

7 Discussion

It is worth comparing our algorithm and results to that of Torralba *et al.* [16]. We have used AdaBoost instead of GentleBoost (used in [16]) as in experiments it gave superior performance and proofed that it is more suitable for our type of weak detectors. Compared to [16] we share significantly fewer entries as they have a 4-fold reduction, compared to our 2-fold reduction. This is mainly caused by their type of basic features which are much less complex and thus more common over different categories than ours.

Initial experiments show that a combination of our model with appearance patches increases the detection performance, but this is the subject of future work.

Class	Plane	CarR	Mb	Face	B-S	B-R	B-F	Car23	CarF	Bottle	CowS	H-S	H-F	CowF	Pers.	Mug	Cup
Ref.	6.3 [5],C	6.1 [10],D	7.6 [15],D	6.0 [15],D							0.0 [10],D						
I,$\mathcal{T}$	7.4	2.3	4.4	3.6	28.0	25.0	41.7	12.5	10.0	9.0	0.0	8.2	13.8	18.0	47.4	6.7	18.8
J,$\mathcal{T}$	7.4	3.2	3.9	3.7	22.4	20.8	31.3	12.5	7.6	10.7	0.0	7.8	11.5	12.0	42.0	6.7	12.5
I,$\mathcal{M}$	1.1	7.0	6.2	1.4	10.3	7.7	8.5	5.2	7.6	7.1	1.6	10.0	8.2	9.5	29.1	5.1	8.0
J,$\mathcal{M}$	1.5	4.3	4.5	1.6	8.9	5.9	7.7	3.8	8.5	6.1	1.3	11.0	4.7	6.8	27.7	5.8	8.3

Table 2: Detection results. In the first row we compare categories to previously published results. We distinguish between detection D (RPC-eq.-err.) and classification C (ROC-eq.err.). Then we compare our model, either trained by the independent method (I) or by the joint (J) method, and tested on the class test set $\mathcal{T}$ or the multiclass test set $\mathcal{M}$. On the multiclass set we count the best detection in an image (over all classes) as the object category. The abbreviations are: B=Bike, H=Horse, Mb=Motorbike, F=Front, R=Rear, S=Side.

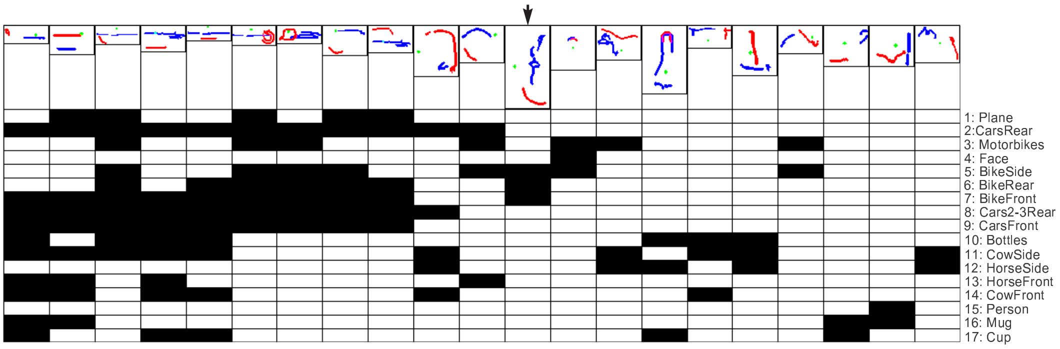


Figure 9: Examples of weak detectors that have been learnt for the whole dataset (resized to the same width for this illustration). The black rectangles indicate which classes share a detector. Rather basic structures are shared over many classes (e.g. column 2). Similar classes (e.g. rows 5, 6, 7) share more specific weak detectors (e.g. column 12, indicated by the arrow, where parts of the bike's wheel are shared).

Acknowledgements

This work was supported by the Austrian Science Fund, FWF S9103-N04, Pascal Network of Excellence and EC Project CLASS.

References

[1] S. Agarwal and D. Roth. Learning a sparse representation for object detection. In *Proc. ECCV*, volume 4, pages 113–130, 2002.

[2] E. Bart and S. Ullman. Cross-generalization:learning novel classes from a single example by feature replacement. In *Proc. CVPR*, volume 1, pages 672–679, 2005.

[3] G. Csurka, C. Bray, C. Dance, and L. Fan. Visual categorization with bags of keypoints. In *ECCV04. Workshop on Stat. Learning in Computer Vision*, pages 59–74, 2004.

[4] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *Proc. CVPR Workshop on Generative-Model Based Vision*, 2004.

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

[6] Y. Freund and R. Schapire. A decision theoretic generalisation of online learning. *Computer and System Sciences*, 55(1):119–139, 1997.

[7] http://www.emt.tugraz.at/∼pinz/data/multiclass/.

[8] http://www.pascalnetwork.org/challenges/VOC/.

[9] S. Krempp, D. Geman, and Y. Amit. Sequential learning of reusable parts for object detection. Technical report, CS Johns Hopkins, 2002.

[10] B. Leibe, A. Leonardis, and B. Schiele. Combined object categorization and segmentation with an implicit shape model. In *ECCV04. Workshop on Stat. Learning in Computer Vision*, pages 17–32, May 2004.

[11] D. Magee and R. Boyle. Detection of lameness using re-sampling condensation and multi-steam cyclic hidden markov models. *IVC*, 20(8):581–594, 2002.

[12] A. Opelt, M. Fussenegger, A. Pinz, and P. Auer. Generic object recognition with boosting. *PAMI*, 28(3), 2006.

[13] A. Opelt, A. Pinz, and A. Zisserman. A boundary-fragment-model for object detection. In *Proc. ECCV*, volume 2, pages 575–588, May 2006.

[14] E. Seemann, B. Leibe, K. Mikolajczyk, and B. Schiele. An evaluation of local shape-based features for pedestrian detection. In *Proc. BMVC*, 2005.

[15] J. Shotton, A. Blake, and R. Cipolla. Contour-based learning for object detection. In *Proc. ICCV*, volume 1, pages 503–510, 2005.

[16] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing features: efficient boosting procedures for multiclass object detection. In *Proc. CVPR*, 2004.