Int J Comput Vis (2014) 106:252–268
DOI 10.1007/s11263-013-0670-8

Branch&Rank for Efficient Object Detection

Alain D. Lehmann · Peter V. Gehler · Luc Van Gool

Received: 14 September 2012 / Accepted: 24 October 2013 / Published online: 24 December 2013

Abstract Ranking hypothesis sets is a powerful concept for efficient object detection. In this work, we propose a branch&rank scheme that detects objects with often less than 100 ranking operations. This efficiency enables the use of strong and also costly classifiers like non-linear SVMs with RBF-χ^2 kernels. We thereby relieve an inherent limitation of branch&bound methods as bounds are often not tight enough to be effective in practice. Our approach features three key components: a *ranking function* that operates on *sets of hypotheses* and a grouping of these into different *tasks*. Detection efficiency results from adaptively subdividing the object search space into decreasingly smaller sets. This is inherited from branch&bound, while the ranking function supersedes a tight bound which is often unavailable (except for rather limited function classes). The grouping makes the system effective: it separates image classification from object recognition, yet combines them in a single formulation, phrased as a structured SVM problem. A novel aspect of branch&rank is that a better ranking function is expected to decrease the number of classifier calls during detection. We use the VOC'07 dataset to demonstrate the algorithmic properties of branch&rank.

Keywords Branch&rank · Object detection · Non-linear kernel classifier · Sub-linear detection

A. D. Lehmann (✉) · L. V. Gool
Computer Vision Laboratory, ETH Zurich, Zurich, Switzerland
e-mail: lehmann@vision.ee.ethz.ch; alain.lehmann@gmail.com

P. V. Gehler
MPI for Intelligent Systems, Tübingen, Germany
e-mail: pgehler@tuebingen.mpg.de

L. V. Gool
ESAT-PSI/IBBT, KU Leuven, Leuven, Belgium
e-mail: vangool@vision.ee.ethz.ch

1 Introduction

Object class detection in images is challenging because of two problems. First, object appearances exhibit large variations due to intra-class variability, illumination changes, etc. Second, objects may appear anywhere in an image with unknown scale, and need to be localised. Hence, detectors have to simultaneously cope with appearance variations and a large search space of possible object positions.

Handling the appearance variations is a crucial factor for good detection accuracy and thus to score well in current object detection benchmarks. Much progress has been made lately, manifesting itself in increasing evaluation scores of the VOC benchmark (Everingham et al. 2007) during the last years. These advances suggest that strong classifiers and combination of different image descriptors are required; non-linear SVMs are constantly found to be well suited for this task (Vedaldi et al. 2009; Gehler and Nowozin 2009). Unfortunately, such classifiers are expensive to evaluate which makes it challenging to master the large search space: sufficiently fine grained sliding window search typically requires >10k classifier evaluations. This imposes a limited computational budget per classifier call that renders non-linear SVMs intractable. This leaves two possible options to handle the search space: (a) reducing the cost of a single classifier evaluation (Viola and Jones 2004; Vedaldi et al. 2009) or (b) reducing the number of classifier calls (Lampert et al. 2009; Lehmann et al. 2011b). Let us stress this point, we distinguish between the cost of the classifier and the number of times it is called. These two factors are orthogonal and their product yields the total runtime (c.f. Fig. 1). We next discuss both options separately, but note that they can also be combined (Lampert 2010; Weiss et al. 2010).

Reducing the classifier cost is a common strategy to deal with the limited computational resources. Cascade classi-

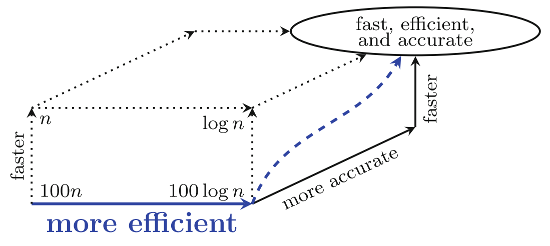

Fig. 1 Computational aspects (i.e., efficiency and speed) are important, although current benchmarks exclusively focus on detection accuracy. *Efficiency* captures an algorithm's scalability to locate an object among n possible hypotheses (expressed with big-O notation). *Speed and accuracy* depend on the choice of the classifier (including its training and used features). We advocate to control all three properties, but to first focus on efficiency as it dominates for growing n (e.g. multi-class setups)

fiers (Viola and Jones 2004; Vedaldi et al. 2009; Felzenszwalb et al. 2010) are prominent examples: they reject many hypotheses with a simple criterion and thereby avoid many computations. However, they do not per-se reduce the number of calls and the total runtime still scales linearly in the number of detection sites. Similarly, one can utilise faster classifiers (Gall and Lempitsky 2009; Zhang et al. 2011a), optimise the implementation (Wei and Tao 2010), or leverage the massive parallelism of GPU architectures (Prisacariu and Reid 2009; Wojek et al. 2008). All these approaches process the search space *exhaustively* and are thus not scalable. In other words, reducing the classifier cost makes a detector *faster* but not *efficient*.

Reducing the number of classifier calls is the alternative method to make detection scalable. Branch&bound (Breuel 2002; Keysers et al. 2007; Lampert et al. 2009; Lehmann et al. 2011b) is a promising approach that falls in this category: it partitions the search space *adaptively* and thereby avoids exhaustive search. Branch&bound operates on sets of hypotheses and uses a bounding function to process the most-promising sets in a best-first order. This allows for sub-linear runtime given a *tight* bound on the classifier function with exhaustive search as worst-case complexity. Unfortunately, bounds are generally not tight enough and the method's efficiency is still insufficient to deploy powerful (and thus often costly) classifiers (c.f. Vedaldi et al. 2009); such classifiers seem however essential to deal with the appearance variations. This indicates that the concept of bounds is a limiting factor and we identify the notion of *sets* as the key to efficiency. This paper sets out to rectify the problem of bounds: branch&rank improves efficiency and therefore enables the use of costly classifiers i.e., rich appearance models.

Branch&rank (Lehmann et al. 2011a) follows the ideas of branch&bound but overcomes its limitations. Specifically, we abandon the notion of bounds and thereby allow for arbitrary classifiers. We adopt the best-first search and "branch" but do not "bound". Instead, we explicitly integrate the idea of scoring sets into the training problem. Intuitively speaking we aim to "learn the bound". More precisely, we learn a ranking function that prioritises hypothesis sets that do *contain an* object over those that do not. This branch&rank scheme is more efficient and detects objects with often less then 100 ranking operations. This enables the use of expensive classifiers. Although we could apply the ranking paradigm to arbitrary functions, we deliberately choose to work with non-linear SVMs and RBF-χ^2 kernels. These classifiers have been shown to perform well (Gehler and Nowozin 2009; Lazebnik et al. 2006; Vedaldi et al. 2009), but are generally perceived as being too slow to be directly applicable. Yet, we show that using *only* evaluations of these non-linear SVMs is feasible. We train them in a multi-task setup which accounts for the size of hypothesis sets; we thereby separate image classification from object recognition, yet combine them in a joint objective.

In summary, the benefits of ranking are the following: (1) The ranking condition combines model estimation and acceleration of the test-time problem in a *joint objective*: improving the ranking function (classifiers) leads to better *and more efficient* object detection. (2) The ranking condition is flexible; it allows for *arbitrary* (ranking-)classifiers since no bound has to be derived. (3) Branch&rank is efficient and enables the use of strong and costly classifiers (like non-linear SVMs) without the need for cascade-like approximations.

In the sequel, Sect. 2 presents related work and Sect. 3 covers the overal branch&rank algorithm. The reason for multiple tasks as well as a SVM-based ranking function are presented in Sect. 4. Various compact set description are detailed in Sect. 5, followed by experiments in Sect. 6.

2 Related Work

Branch&Bound. Branch&bound for bounding box detection (Lampert et al. 2009) pioneered the field of efficient object detection; the number of classifier calls of such methods scales sub-linearly with the search space size. Its efficiency depends on the availability of a tight bounding function, that, unfortunately, is only available for very small function classes. The obvious bounds for non-linear SVMs are simply not sufficiently tight to be of practical value. This severely limits the use of this technique for the task of object detection, that requires strong classification functions. Approximate bounds which are tighter are shown to accelerate convergence (Lehmann et al. 2011b). But again, this was only shown for simple function classes. Moreover, Lehmann et al. (2011b) give up on global optimality guarantees, a much appreciated property of branch&bound (Lampert et al. 2009). We will elaborate on optimality and discuss the property of our method in Sect. 3.5.

Coarse- to- Fine. Coarse-to-fine detectors (Gangaputra and Geman 2006; Pedersoli et al. 2010) also operate on sets. However, they start with a uniform partitioning at the coarsest level that still scales linearly (similar to a sliding-window approach); only the finer levels partition subsets in an adaptive way. Gangaputra and Geman (2006) use a cost-to-power criterion to learn how to partition a given set. They examine potential sets in a breadth-first order and *prune non-promising sets*. Pedersoli et al. (2010) refine a set uniformly and propagate only the locally best-scoring hypothesis to the next finer level. This local non-maximum suppression focuses on the most promising hypothesis (within a neighbourhood) and *prunes all others*. In contrast, we start with the entire search space, partition a set and explore the globally best option: We *prioritise promising sets* and we *never prune*. In fact, the concept of pruning is closely related to the idea of cascade detectors.

Cascades. Cascades have been used with much success to reduce the computation cost of classifiers (Viola and Jones 2004; Vedaldi et al. 2009; Felzenszwalb et al. 2010). They use simple criteria to reject many hypotheses and thereby reduce the number of strong classifier evaluations. However, they still process *every* bounding box exhaustively. In other words cascades are fast but not efficient. Although the cascading in (Felzenszwalb et al. 2010) does not examine every possible *part configuration*, the root part (which is reported as detection) is evaluated exhaustively as in a sliding-window approach.

Adaptive Cascades. Furthermore, cascading and adaptive sub-division are two orthogonal techniques (Lampert 2010; Weiss et al. 2010). Structured ensemble cascades (Weiss et al. 2010) leverage the coarse-to-fine approach and only subdivide hypothesis sets that have not yet been filtered. Again, they prune whereas we prioritise sets. Moreover, they work on pose estimation and focus on localising part configurations; their images are "largely focused on a single actor" and thus avoid the object localisation that we tackle in this work. Lampert (2010) is more closely related to our approach as it does best-first search and it presents a cascade of bounds. The bounding still limits the possible functions to be used and only the final cascade stage is non-linear; our approach uses non-linear SVMs throughout the entire search.

Candidate Proposals. Other approaches reduce the number of classifier calls by proposing possible bounding boxes that can subsequently be verified using an object detector of choice. For example, Chum and Zisserman (2007), Razavi et al. (2011) generate class-specific proposals based on discriminative visual words while Alexe et al. (2010) propose class-independent object positions using various low-level saliency measures. In object segmentation similar techniques are applied with much success (Carreira and Sminchisescu 2010). Such two-step proposal-verification schemes are effective but lack a joint objective function. Thus they are difficult to optimise and the influence of each single part on the entire system is not trivial to measure. Saliency-based attention methods (Itti et al. 1998; Alexe et al. 2010) actually reason in a bottom-up fashion driven by low-level cues. In contrast, our ranking algorithm guides its attention based on high-level hypotheses.

Context. Context is a valuable source of information that many object detectors try to exploit; so does branch&rank. A crucial difference w.r.t. other approaches is that they focus on detection accuracy, while we address efficiency. The work by Blaschko and Lampert (2009) is closest to ours as they combine global and local context in an SVM setup. Their aim is however to improve the detection confidence and it is unclear if this positively affects the convergence of their branch&bound approach. Moreover, they do not consider the continuum between global and local context as we do. Object priming (Torralba 2003) is another prominent example that leverages global context: they capture the "gist" of a scene (Oliva and Torralba 2001). Work along this line (Torralba 2003; Murphy et al. 2003; Bileschi and Wolf 2005; Torralba et al. 2010; Harzallah et al. 2009) has shown that such holistic image features boost detection accuracy, but computational aspects were not considered or improved. Finally, Wolf and Bileschi (2006) argue that context is particularly helpful to detect "difficult" objects (e.g., small, low-resolution instances) but not so much in the general case. They measured only *accuracy* though and branch&rank suggests that context helps to improve detection *efficiency*.

3 Branch and Rank

3.1 Overview

Efficient detection means to find objects without examining all possible hypotheses individually. This is possible as many hypotheses are strongly correlated e.g., due to overlapping bounding boxes or similarity of classes. These can be grouped into *sets* and processed as one single entity. We first focus on the core algorithm and thus postpone a detailed discussion of sets to Sect. 5; the overall idea is sketched in Fig. 2.

Branch&rank (Lehmann et al. 2011a) aims to *focus on sets that contain objects*, rather than spending computation on sets that do not. The detector iteratively splits such sets to eventually identify a single bounding box. The challenge is to accurately decide which sets contain an object—and should thus be split—without examining every set member individually. To this end, we classify the sub-image that covers the union of every bounding-box of a set. This is a challenging task and we thus want to avoid making hard decisions that would reject hypotheses prematurely. Therefore, we keep a partitioning of the search space (represented by sets) and refine (i.e., split/branch) sets that most likely con-

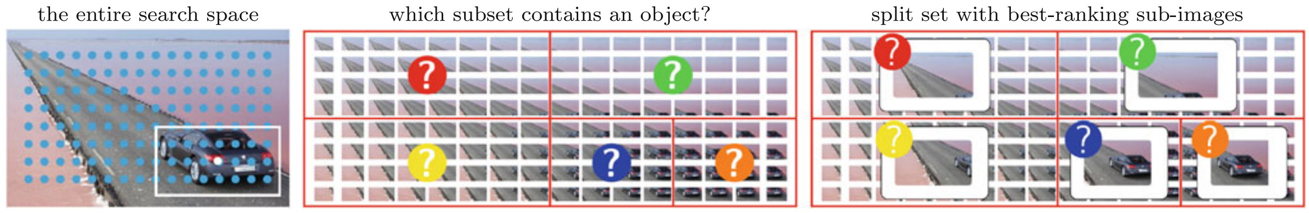
the entire search space
which subset contains an object?
split set with best-ranking sub-images

Fig. 2 *Left* branch&rank aims to *efficiently* find bounding boxes containing an object *white* among all possible hypotheses. For illustrative reasons let us assume all hypothesis are only those depicted by the *blue dots*. *Middle* the algorithm uses *hypothesis sets red rectangles* to partition the large search space (here 8 × 16 hypotheses). Efficient detection results from *refining sets that probably contain objects*. *Right* branch&rank *classifies the sub-image* that covers all bounding boxes of a set to decide which set to refine next (Color figure online)

Table 1 Notation

We distinguish single hypothesis λ, hypothesis sets Λ, and sets of sets $\mathbb{L}$ by different letters. An additional $^+$ superscript indicates that at least one positive annotation is contained

λ	Parametrisation of a *single* bounding box
Λ	Parametrisation of a *set* of bounding boxes
$\mathbb{L}$	*The set* of *all sets* of bounding boxes
$\mathbb{L}^+ \subset \mathbb{L}$	The set of all sets *containing at least one object*
$\Lambda_j^+ \in \mathbb{L}^+$	A set *containing at least one annotated object*
$Y = \{\lambda_i^+\}_1^m$	The training data: m ground truth annotations λ_i^+
$B(\lambda)$, $B(\Lambda)$	(Four corner) bounding box for parametrisation λ and sets Λ
$\phi(\Lambda)$	Appearance descriptor for hypothesis set Λ
$\Delta : \Lambda \times \Lambda \mapsto \mathcal{R}$	(Set-valued) loss function for pair of examples
$f : \Lambda \mapsto \mathcal{R}$	Ranking function: provide a *priority* for hypothesis set Λ
f^{LA}, f^{GT}	Loss-augmented and ground truth score function
$q : \Lambda \mapsto \{1, \dots, T\}$	Task mapping to distinguish T different tasks.
w_t, b_t	SVM weight vector and bias term for task t (with $t = q(\Lambda)$)

tain an object. This search strategy traverses the hypotheses set in a order that visits highest scoring elements first. In combination with applying a detection threshold we can thus avoid having to compute scores for all possible bounding boxes in a principled way.

The efficiency of branch&rank results from splitting sets that probably contain objects *first*, which naturally leads to a ranking problem. This section details the theoretical background as well as the overall algorithm, while the next section gives a concrete implementation.

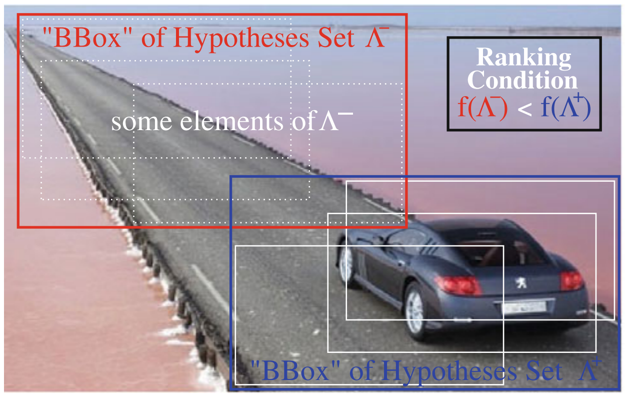

Fig. 3 The ranking condition: hypothesis set containing at least one object *blue* should be ranked higher than sets not containing any objects *red*. Perfect ranking results in logarithmic detection time; in practice, we find objects with often less than 100 ranking operations (Color figure online)

3.2 Ranking Condition

The core of our detector is a ranking function. Its goal is to order a priority queue: sets containing true detections should be ranked higher than all others. More formally, let $\mathbb{L}$ be the set of all hypothesis sets while $\mathbb{L}^+$ represents only those that contain at least one object. The ranking condition then reads as

$$f(\bar{\Lambda}) < f(\Lambda^+) \qquad \forall \Lambda^+ \in \mathbb{L}^+, \quad \forall \bar{\Lambda} \in \mathbb{L} \backslash \mathbb{L}^+, \tag{1}$$

where the set Λ^+ contains at least one object and $\bar{\Lambda}$ contains no object; our notation is summarised in Table 1 while Fig. 3 illustrates the ranking condition. Learning a function which aims to fulfill this condition is the topic of the next section. Let us now assume we had access to a function f^{oracle} that meets condition (1) for any observable image. In that case, branch&rank first examines all sets that do contain an object and the size of these sets decreases exponentially fast (as we always split them into equal-sized subsets). A perfect ranking function thus implies logarithmic detection time.

Algorithm 1 Detector(ranking function f, image I)

```
D = $\emptyset$
priorityqueue.enqueue ($\infty$, $\Lambda^0(I)$)
while True do
    score, $\Lambda$ = priorityqueue.dequeue_best()
    if score $\leq \tau$ then return D
    elif $minOL(\Lambda, D) > 0.5$ then continue
    elif $|\Lambda| < \varepsilon$ then $D = D \cup \Lambda$
    else for $\Lambda_i$ in split($\Lambda$) do
        priorityqueue.enqueue($f(\Lambda_i)$, $\Lambda_i$)
```

3.3 Best-First Search

Algorithm 1 summarises the adaptive, best-first subdivision strategy. The search is governed by a priority queue each element of which represents a set of hypotheses; the ordering is according to the *score* of a ranking function. The algorithm proceeds as follows. Initially, the whole search space $\Lambda^0(I)$ is entered into the queue as a single element. Subsequently at each step, the highest ranking element in the priority queue is split into smaller subsets; they are subsequently scored and inserted into the priority queue. In case the highest ranking hypothesis set is a single bounding box (or a sufficiently small set) a detection is reported. This results in a kD-tree partitioning of the entire search space. We detect multiple instances by means of non-maximum suppression. In contrast to commonly used sliding-window search, we have to suppress already during the iterative splitting rather than during a post-processing step. This is detailed in the next subsection.

3.4 Non-maximum Suppression

An image may contain multiple object instances and we would like to detect them all. Moreover we should avoid re-detecting a given object twice as this counts as false positive. Re-detections are usually eliminated with a non-maximum suppression post-processing, but a best-first search algorithm cannot wait for the post-processing: It would re-detect the same instance over and over again and loose its sub-linear runtime. We elude this problem with a simple suppression scheme, while more elaborate approaches exist (Blaschko 2011; Desai et al. 2009).

Our best-first algorithm aims to detect local optima and we assume that the first detection would survive a non-maximum post-processing. Hence, we can directly suppress or penalize hypotheses that would get suppressed by this detection. However, to maintain the efficiency of the algorithm, we have to adapt the non-maximum suppression to *sets* of hypotheses.

The usual non-maximum suppression eliminates any hypothesis λ that has a significant bounding-box $B(\lambda)$ intersection-over-union overlap (Everingham et al. 2007)

$$ol(\lambda, \lambda_i) = \frac{area(B(\lambda) \cap B(\lambda_i))}{area(B(\lambda) \cup B(\lambda_i))} \tag{2}$$

with a higher scoring detection λ_i. Consequently, we suppress a set if *all* of its elements exceed a certain threshold. Therefore, what matters is the *minimal* overlap

$$minOL(\Lambda, D) = \max_{\lambda_i \in D} \min_{\lambda \in \Lambda} ol(\lambda, \lambda_i) \tag{3}$$

with any previous detections $\lambda_i \in D$. We then suppress a set Λ if $minOL(\Lambda, D) > 0.5$.[1] This simple modification allows for detecting multiple object instances. Although this step suppresses certain hypotheses, the overall best-first search should not be thought of a pruning method (as e.g., detection cascades). The reason is that all objects contained in suppressed sets would be eliminated by a usual non-maximum suppression post-processing. Therefore, there is no need to keep them in the priority queue.

3.5 Connection to Branch&Bound

A connection to branch&bound (Lehmann et al. 2009; Lampert and Blaschko 2009) is evident from the name; this subsection comments on the relationship in more detail. First of all, note that the only algorithmic difference is to replace the ranking function while everything else remains unchanged. Branch&bound prioritises sets by an upper bound $\hat{g}(\Lambda) \geq \max_{\lambda \in \Lambda} g(\lambda)$ to a traditional bounding-box score function $g(\lambda)$. This leads to different properties and guarantees that we want to elaborate.

Assumptions and Requirements. Branch&bound requires that there exists a tight upper-bound for a given score function. In contrast branch&rank assumes that it is possible to efficiently decide whether a set (in our case represented by a sub-image) *contains an object* or not. In fact, image classification addresses exactly this problem and it does so with much success (Everingham et al. 2007). This suggests that the assumption is valid in practice. Of course, it also suggests that the detection strategy is challenged when objects are presented out-of-context (e.g., a car in a kitchen). However, it seems that this assumption is rather benign when compared to branch&bound's requirement of a *tight* bound (Vedaldi et al. 2009).

On Convergence. We discussed that branch&rank has logarithmic detection time in case of a perfect ranking function. This is in contrast to branch&bound where the efficiency depends on the quality of the bound, which is unrelated to the generalisation problem. In other words, given a perfect bounding-box classifier g the runtime of branch&bound with bound $\hat{g}$ can still be linear time (e.g., $\hat{g}(\Lambda) = \infty$ except for $\hat{g}(\lambda) = g(\lambda)$). Of course, we cannot expect perfect general-

[1] To avoid a hard decision, one may subtract a penalty from the score and re-schedule the element in the priority queue

isation in practice and every incorrect ranking increases the number of iterations till detection. But we conclude that the ranking condition couples accuracy and efficiency. This suggests that the better a ranking function, the better *and more efficient* the detector.

On Optimality. We follow Bottou and Bousquet (2008) and distinguish the following types of errors. The *approximation error* which is due to the function class we search a classifier in and the *test-time error* which is incurred by only approximately solving the test-time search problem. We neglect the third possible cause of an error, which is the optimization error. This type of error results from solving learning problems approximately. For branch&bound as in (Lampert et al. 2009) the test-time error is zero, but the approximation error is high due to the limited function classes for which tight bounds are known. Our approach incurs a test-time error because we can not guarantee finding the best scoring bounding box, but the approximation error decreases over branch&bound because we can use richer function classes, e.g., non-linear SVMs. The recent research on object detection (e.g. Vedaldi et al. 2009) indicates that the decrease of the approximation error outweighs the test-time error. However, the optimization problem that we formulate next aims to actively reduce the test-time error. A lower objective yields a better ranking that should increase accuracy and speed up convergence, both at the same time.

4 Multi-task SVM Ranking

This section presents a concrete ranking function along with its training procedure. We aim for using non-linear SVMs as they are constantly found to cope well with appearance variations of image scenes and object classes (Vedaldi et al. 2009; Gehler and Nowozin 2009). However, the branch&rank paradigm is general and can also be implemented with other classifiers (e.g., random forests, boosting, etc).

Section 4.1 introduces the ranking function and motivates the idea of grouping hypothesis sets into multiple tasks. Subsequently, Sect. 4.2 revisits a structured SVM formulation dedicated to learning rankings. We then present a transformation of the problem in Sect. 4.3 that decomposes the training. The resulting optimisation problem is stated in Sect. 4.4. Finally, Sect. 4.5 elucidates locally linear SVMs (Zhang et al. 2011b) to speed-up the evaluation.

4.1 Multi-task Ranking Function

Bag-of-Words Appearance. Our ranking function builds on the commonly used bag-of-words approach as it copes well with image classification and object detection (Lazebnik et al. 2006; Vedaldi et al. 2009; Lampert et al. 2009; Lehmann et al. 2011b). This aggregates (local) features and

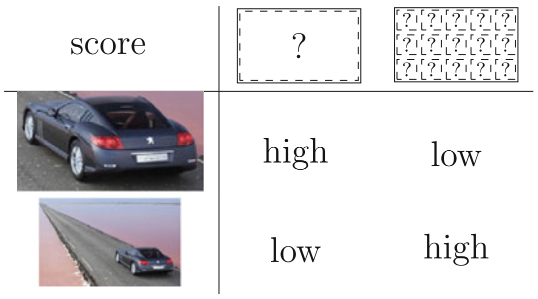

Fig. 4 Ranking Ambiguity. One large bounding-box *left* and a set of many small bounding-boxes *right* can cover the same sub-image. Their semantics is different though i.e., is the object localised *top* or does it contain an object somewhere *bottom*. Therefore, a ranking function that ignores the set size would have to predict two different ranks for the same descriptor, which is impossible

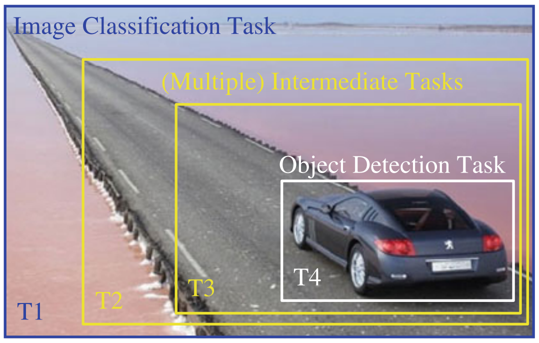

Fig. 5 Detecting an object in an image is decomposed into different tasks. The approach smoothly blends from image classification (T1) to object detection (T4). This allows for object localization with runtime sub-linear in the number of candidate regions, often using less than 100 classifier evaluations

represents them by a histogram. Specifically, the appearance descriptor $\phi(\Lambda)$ uses all features that fall within the bounding box $B(\Lambda) := \bigcup_{\lambda \in \Lambda} B(\lambda)$ where the union extends the notion of a bounding box to sets of hypotheses (c.f. Fig. 3). For large sets, this actually includes all image features as usual in image classification.

Appearance is Not Enough. Figure 4 illustrates a degenerate case which suggest that the appearance within the bounding-box union is not sufficient to properly rank a set. In fact, using only $\phi(\Lambda)$ may lead to ambiguities that jeopardize the ranking. In short, two sets with *different labels* can yield the same bounding box union; thus the *same appearance descriptor*. We address this problem with a multi-task framework that connects image classification with object detection.

From Classification to Detection. Figure 4 suggests that using *one* ranking function for all possible sets is not sufficient. Let us consider the two extremes of such sets that arise during the search (c.f. Fig. 5).

At one end, the initial set represents the entire image and all possible sub-windows. Scoring this set is the task of image classification. The other extreme is a hypothesis set with only one instance, corresponding to scoring a single bounding box. This is an object recognition problem. Both of course are related, but note the difference in the tasks: the first set should have a high score if it *contains an object*, the latter if it is *centered on the object*. This suggests that these tasks are better solved separately *but* combined in a joint objective. For example the first task could benefit from different image features such as the gist of a scene (Oliva and Torralba 2001), while the latter could make use of object specific features (Oliva and Torralba 2001; Dalal and Triggs 2005; Lowe 2004) or spatial configurations of object parts (Felzenszwalb et al. 2008). In our experiments we did not take advantage of size dependent image representations but focus more on the algorithmic properties of the systems. Moreover, grouping related examples into tasks reduces the intra-task variability, which simplifies the learning problem.

Multi-task Mapping. We capture the notion of tasks by a mapping $q(\Lambda) \mapsto \{1, 2, \ldots, T\}$ which assigns a task ID to any hypothesis set. This mapping builds on properties other than a set's appearance. In other words, the set provides valuable *domain knowledge*. For example, we can use the number of bounding boxes contained in a set (i.e., its size $|\Lambda|$) to differentiate between the classification and recognition tasks. More specifically, our task mapping discretizes $\log(|\Lambda|)$ uniformly into T different tasks; the log accounts for the set size's exponential decay (due to the splitting scheme). This spans the continuum between image classification and the final object recognition problem. While both extremes are often dealt with separately (Griffin et al. 2007; Everingham et al. 2007), we combine and complement them with intermediate tasks. Unless stated otherwise, we will use $T = 6$ tasks. Let us stress that this is one particular choice, while the concept of tasks is more general: we could define other mappings that group examples by scale and aspect-ratio (Park et al. 2010; Zhang et al. 2011b), or also by class labels (Yeh et al. 2009).

Ranking Function. Finally, we define the multi-task ranking function as

$$f(\Lambda) = \langle w_{q(\Lambda)}, \phi(\Lambda)\rangle + b_{q(\Lambda)} \tag{4}$$

with per-task weight vectors w_t and bias terms b_t, where $t = q(\Lambda)$ designates the task of sample Λ. In a kernelized form this relates to a product kernel $\mathbb{I}_{q(\Lambda)=q(\Lambda')}\phi(\Lambda)^T\phi(\Lambda')$ with indicator function $\mathbb{I}_x$ (that is 1 if x is true and 0 otherwise) when dualising the SVMs. The SVM optimisation will eventually benefit from the per-task biases as they lead to a decomposition that allows for training tasks in isolation. For simplicity, we use the same appearance descriptors for all tasks, although one could specifically tailor them to the individual tasks possibly making use of different features.

4.2 Structured SVM Ranking

Several ranking problems have been studied in the machine learning literature (Tsochantaridis et al. 2005; Chapelle and Keerthi 2009; Burges et al. 2005) and we adopt structured SVMs using margin-rescaling (Tsochantaridis et al. 2005).

We briefly revisit the optimisation problem and refer to the original paper for details. In this framework, the ranking condition from the previous section translates to the optimisation problem

$$\min_{f,\xi\geq 0} \|f\|^2 + C\sum_{j=1}^{n}\xi_j \tag{5}$$
$$\text{s.t. } f(\Lambda_j^+) - f(\Lambda) \geq \Delta(\Lambda_j^+, \Lambda) - \xi_j$$
$$\forall \Lambda_j^+ \in \mathbb{L}^+, \qquad \forall \Lambda \in \mathbb{L}\backslash\mathbb{L}^+$$

with slack variables ξ_j for every positively annotated example $\{\Lambda_j^+\}_{j=1}^n$ and regularisation parameter C, that trades data fit for model complexity ($\|f\|^2 = \sum_{t=1}^T \|w\|^2$). The loss $\Delta(\Lambda_1, \Lambda_2) \mapsto \mathbb{R}$ encodes the cost of predicting Λ_2 if Λ_1 were correct. As we deal with object *class* detection (where specific instances are not distinguished), we need to properly handle the case of multiple objects in an image: detecting an object λ_1^+ instead of object λ_2^+ should not incur any loss, i.e., $\Delta(\lambda_1^+, \lambda_2^+) = 0$ (and similar for sets Λ^+). In other words, the loss depends on all annotations Y.

We adopt and extend the loss introduced in (Blaschko and Lampert 2008). They defines the loss of a bounding box proportional to its overlap with the ground truth; high overlap means small loss and vice versa. We extend this loss to sets by defining the overlap of hypothesis sets as

$$max\,OL(\Lambda, Y) = \max_{\lambda_i\in Y}\max_{\lambda\in\Lambda} ol(\lambda, \lambda_i) \tag{6}$$

with ol as defined in Eq. (2); this uses the maximal rather than the minimal overlap in Eq. (3). Hence, $max\,OL$ is high if one instance (of the set Λ) has high overlap with any of the ground truth objects and 0 if there is no instance with any overlap. Finally, we define the loss as $\Delta(\Lambda) := \Delta(\Lambda_j^+, \Lambda) = 1 - max\,OL(\Lambda, Y)$ which exploits the fact that during training the first argument is always a positive example. The latter yields the "1−" term which could otherwise be a "max" term similar to the second one. Note that this loss is identical to (Blaschko and Lampert 2008) if there were no sets (in Eq. (5)), but only single bounding boxes.

4.3 Problem Decomposition

The multi-task ranking function from the previous subsection allows for a decomposition of the SVM optimisation problem. This reduces the complexity since each of the resulting problems involves only a subset of the data. This Subsection

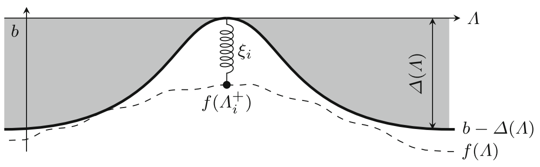

Fig. 6 Constraints of the decoupled structured SVM as in Eq. (7). The classifier/ranking-function f *dashed line* is constrained to be below the shaded region. The slack variables ξ_i act like springs and force the positive examples Λ_i^+ to have a higher rank

details the two necessary steps to accomplish this decomposition.

Step 1: Constraint Decoupling. We exploit the special structure of our loss function to break the pairwise dependencies in Eq. (5).[2] More specifically, we rewrite the constraint set from $f(\Lambda_j^+) - f(\Lambda) \geq \Delta(\Lambda) - \xi_j$ to $f(\Lambda_j^+) + \xi_j \geq f(\Lambda) + \Delta(\Lambda)$ which, of course, must still hold $\forall \Lambda_j^+ \in \mathbb{L}^+, \ \forall \Lambda \in \mathbb{L}\setminus\mathbb{L}^+$. The main observation is that each side now involves only Λ *or* Λ_j^+. Moreover, each side provides a lower/upper bound for the other. To be precise, there exists a value b for which $f(\Lambda_j^+) + \xi_j \geq b$ and $b \geq f(\Lambda) + \Delta(\Lambda)$. We therefore obtain an equivalent optimisation problem

$$\begin{aligned} \min_{f,\xi\geq 0,b} \quad & \|f\|^2 + C\sum_{j=1}^{n}\xi_j, \\ \text{s.t.} \quad & f(\Lambda_j^+) + \xi_j \geq b \qquad \forall \Lambda_j^+ \in \mathbb{L}^+ \\ & b \geq f(\Lambda) + \Delta(\Lambda) \quad \forall \Lambda \in \mathbb{L}\setminus\mathbb{L}^+ \end{aligned} \tag{7}$$

that explicitly optimises over the value b of these upper/lower bounds. This step transformed the pairwise constraints from the initial problem into constraints which only depend on one element. This resembles much the problem of binary SVMs, except that the constraints for negative examples are not allowed any slack. A visualisation of the decoupled constraints is given in Fig. 6. Let us emphasise that the feasible set of Eqs. 5 and 7 are identical. The latter is however advantageous as it involves much fewer constraints: it has $n + m$ constraints (with $n = |\mathbb{L}^+|$ and $m = |\mathbb{L}\setminus\mathbb{L}^+|$) while the original problem had $n \times m$. In summary, the decoupling simplifies the description of the feasible set, but without changing the optimal solution.

Step 2: Task Decomposition. This second step benefits from the multi-task function Eq. (4) that includes per-task bias terms. Substituting Eq. (4) into the optimisation problem Eq. (7) yields

$$\min_{\substack{\{w_t,b_t\}_{t=1\ldots T}\\ \xi\geq 0,b}} \quad \sum_t \|w_t\|^2 + C\sum_j \xi_j \quad \text{subject to}$$
$$\begin{cases} \langle w_{q(\Lambda_j^+)}, \phi(\Lambda_j^+)\rangle + b_{q(\Lambda_j^+)} \geq b - \xi_j & \forall \Lambda_j^+ \in \mathbb{L}^+ \\ \langle w_{q(\Lambda)}, \phi(\Lambda)\rangle + b_{q(\Lambda)} \leq b - \Delta(\Lambda) & \forall \Lambda \in \mathbb{L}\setminus\mathbb{L}^+ \end{cases} \tag{8}$$

which can be almost decomposed: solely the variable b induces a coupling between the tasks. We further observe that the constraints depend only on the differences $b_t - b$. Therefore, adding a fixed constant to all b_t and b does not affect the solution. In other words, the problem is underconstrained and we choose the additional constraint $b = 1$ to make the optimal solution unique. This eliminates the intertask coupling and it remains to group all constraints and all summands of the objective by their task ID.

Doing so reveals the final optimisation problems that we state in the following section. Let us point out that we did not alter the constraint set that enforces the ranking condition (1).

4.4 Decomposed SVMs and Training

This section states the final optimisation problem and discusses the training procedure. The preceding subsections detailed the SVM setup, multi-task consideration, and a transformation which led to a decomposition of the problem. This enables us to learn each task separately: the optimal per-task parameter pair (w_t, b_t) is obtained by solving smaller optimisation problems

$$\min_{w_t,b_t,\xi\geq 0} \|w_t\|^2 + C\sum_j \xi_j \quad \text{subject to}$$
$$\begin{aligned} \langle w_t, \phi(\Lambda_j^+)\rangle + b_t &\geq 1 - \xi_j & \forall \Lambda_j^+ \in \mathbb{L}^+,\ q(\Lambda_j^+) = t \\ \langle w_t, \phi(\Lambda)\rangle + b_t &\leq 1 - \Delta(\Lambda) & \forall \Lambda \in \mathbb{L}\setminus\mathbb{L}^+,\ q(\Lambda) = t \end{aligned} \tag{9}$$

that use only examples from one given task t.

Although this objective function based on the aforementioned decompositions does allow for separate training, it does not mean that the functions $\langle w_t, \cdot\rangle$ are independent. The problem Eq. (9) is indeed equivalent to Eq. (5) when using f as in Eq. (4). This also results in the scores being calibrated, the same regularization and loss rescaled margin is being used for all of them. In some sense this is only possible because ranking is enforced between two sets where one does contain at least one correct hypothesis. This allows for the constraint decoupling described above. There is no distinguishing between two different sets that do not contain correct hypotheses nor two sets that both do. It would be conceivable to rank smaller sets higher than larger sets which would break this decomposition. This problem can be readily kernelized and we choose to work with the RBF-χ^2-kernel $\langle x, z\rangle_k = k(x, z) = \exp\left(-\gamma \sum_l \frac{(x_l - z_l)^2}{x_l + z_l}\right)$ as an example. As bandwidth γ, we use the inverse of the kernel matrix's median.

[2] More generally, this transformation is possible whenever the loss is separable in its two arguments, i.e., $\Delta(\Lambda^+, \Lambda) = u(\Lambda^+) - v(\Lambda)$ for some functions u and v.

Constraint Generation. We solve Eq. (9) using SVM^{struct} (Tsochantaridis et al. 2005) and delayed constraint generation since the constraint set is huge; it consists of all sets of bounding boxes. We initially generate the positive constraints by running Algorithm 1 using the *ground truth ranking* $f^{GT}(\Lambda) := \mathbb{I}_{\Lambda \cap Y \neq \emptyset}$; let us emphasise that branch&rank uses exactly the same annotation as any other detection approach. Thereafter, we alternate between optimising Eq. (9) with the reduced constraint set, and gathering new examples that violate the constraints. We identify new constraints by running a detector that uses the current estimate of the *loss-augmented score* $f^{LA}(\Lambda) := f(\Lambda) + \Delta(\Lambda)$, and subsequently add them to the constraint set. More precisely, in our implementation we perform 10 rounds in each of which we generate new constraints from 300 randomly chosen training images.

Hard Negative Mining. The newly gathered examples are in fact those that are easily confused with positive ones. They are often called *hard negatives* and delayed constraint generation thus provides a formal justification to the commonly used "hard negative mining" (Dalal and Triggs 2005). This connection was first demonstrated by Blaschko and Lampert (2008) and their extension (Blaschko et al. 2010) addressed the problem of pair-wise constraints in Eq. (5). Our decomposition in Sect. 4.3 breaks the pair-wise coupling and the resulting optimisation problem Eq. (9) makes the connection to the binary SVM setup more explicit. But recall, Eqs. (7–9) are equivalent to the well establish ranking formulation Eq. (5).

4.5 Linearization: Anchor Plane SVMs

We further experiment with locally linear SVMs (Ladický and Torr 2011; Zhang et al. 2011a) to speed-up the evaluation of various detector configurations. The evaluation time of non-linear SVMs scales in the number of support-vectors, which tends to grow linearly with the training data size. This is a downside especially since more training data often leads to better classifiers. Consequently, non-linear SVMs become slower as they become better. A possibility to overcome the computational bottleneck is to work with locally linear SVMs (Ladický and Torr 2011; Zhang et al. 2011a). The rationale is that linear SVM are often too simplistic while non-linear SVMs are too expensive to evaluate; locally linear SVMs aim for combining the advantages of both by assuming that the decision boundary is locally linear. In essence, a local-linear SVM is a linear SVM after mapping each feature x to $\psi(x) = x \otimes \gamma(x)$ with a local coordinate coding function $\gamma : x \mapsto \mathcal{R}^D$; the parameter D increases the capacity of the classifier by expanding the original descriptor $x \in \mathcal{R}^N$ to $N \cdot D$ dimensions. This actually induces a kernel $k(x, z) = (x^T z) \cdot (\gamma(x)^T \gamma(z))$ that compares only examples that have similar mappings.

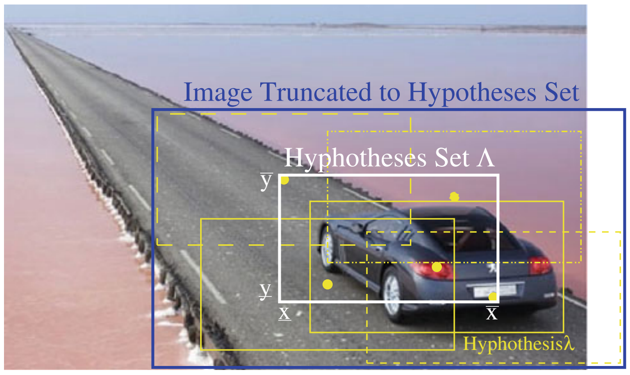

Fig. 7 A hypothesis set *white* with 5 of its elements *yellow*. These are fixed aspect-ratio/scale bounding boxes whose center *dots* are within a fixed interval. All features that fall into the hypothesis set's bounding box *blue* are used to compute a bag-of-word descriptor for Λ (Color figure online)

In particular we adopt the anchor plane SVMs (Zhang et al. 2011a) that use an orthogonal coding function

$$\gamma(x) = v/\|v\|_1 \text{ with } v = Cx \tag{10}$$

with anchor planes defined by a matrix $C \in \mathcal{R}^{D \times N}$ where N denotes the dimensionality of the feature vector x. The anchor planes are found as the dominant eigenvectors of the training-data feature vectors and are found with the singular value decomposition; for details see (Zhang et al. 2011a). In fact, v relates to a PCA approximation of the example x, except that the data mean is not subtracted. Our experiments show that $D = 20$ anchor planes boost the performance compared to linear SVMs. Moreover, the overall evaluation is much faster than non-linear SVMs, in particular, independent of the training data size.

5 Hypothesis Set Representation

This section details the representation of hypothesis sets and we describe how to implement the operations required by the detection algorithm. The overall procedure to represent hypothesis sets is as follows. First, we parametrize bounding boxes using a four dimensional vector. In Sects. 5.1 and 5.2 we work with a reference point plus the box's scale&aspect-ratio, while in Sect. 5.3 the width&height is used instead. Secondly, we obtain sets of bounding boxes by representing each dimension with an interval, rather than a single number. This provides a compact description of sets as illustrated in Fig. 7. The remainder of this section details the representations and set splitting, as well as how to compute the set size and union bounding box.

5.1 Position, Scale, and Aspect-Ratio

Representation & Initialisation. Our first implementation represents a bounding box by its center (x, y), scale

s, and aspect-ratio r. Using (s, r) rather than the box's width/height w, h has the property that it allows to directly control the aspect-ratio of detectable objects. Moreover, encoding scale explicitly relates to the scale-space pyramid as used in many sliding-window systems. More precisely, we work with $ls := \log\sqrt{wh}$ and $lr := \log\sqrt{w/h}$ where the logarithm accounts for the multiplicative nature of the two quantities. Following the overall procedure we represent hypothesis sets as $\Lambda = [\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}] \times [\underline{ls}, \overline{ls}] \times [\underline{lr}, \overline{lr}]$ which comprises all bounding boxes with center $(x, y) \in [\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}]$, and scale/aspect-ratio in the intervals $s \in [\exp(\underline{ls}), \exp(\overline{ls})]$ and $r \in [\exp(2\underline{lr}), \exp(2\overline{lr})]$, respectively. In summary, a single bounding box is represented by a point while sets correspond to hypercubes as illustrated in Fig. 7. The search space $\Lambda^0(I)$ for image I of dimensions $W \times H$ is

$$[0, W] \times [0, H] \times [\log s_{\min}, \log\sqrt{WH}] \times \left[\frac{\log r_{\min}}{2}, \frac{\log r_{\max}}{2}\right]$$

where we estimate the aspect-ratio extremes $r_{\min,\max}$ from the training data, and set the lower scale limit to $s_{\min} = 50$ pixels.[3]

Bounding Box & Set Size. We start by computing a the widest/tallest bounding box of a set, i.e., $w_{\max} = \exp(\overline{ls} + \overline{la})$ and $h_{\max} = \exp(\overline{ls} - \underline{lr})$, respectively. Hence, the rectangle $B(\Lambda) = \left[\underline{x} - \frac{w_{\max}}{2}, \underline{y} - \frac{h_{\max}}{2}, \overline{x} + \frac{w_{\max}}{2}, \overline{y} + \frac{h_{\max}}{2}\right]$ covers all bounding boxes of a given set. Moreover, we define the cardinality of a set as the product of its per-dimension interval sizes (i.e., the upper minus the lower limit). More precisely, we define

$$|\Lambda| = \frac{(\overline{x} - \underline{x})}{\underline{s}} \frac{(\overline{y} - \underline{y})}{\underline{s}} (\overline{ls} - \underline{ls})(\overline{lr} - \underline{lr}) \tag{11}$$

where we normalize the spatial intervals by the smallest scale of the set $\underline{s} = \exp(\underline{ls})$. This ensures scale-invariance as we measure the spatial intervals relative to the object size; we choose the smallest scale since small-scale bounding boxes dominate the size of a set.

Splitting. We implement the splitting scheme as follows. The largest of all four intervals (defining a set) is split into two equals halves. For this we normalize the size of the spatial (xy) intervals using the largest scale $\overline{s}$. This scale-adaptation avoids localising objects with unnecessary high precision: xy-intervals may already be small w.r.t. the largest bounding box of a set; therefore we prefer splitting along ls, lr over spatial splits.

[3] Our visual-words based image descriptors seems inadequate for scales <50 pixels; unfortunately, this yields an a priori loss of recall. This is a problem of the feature representation, not of the detector.

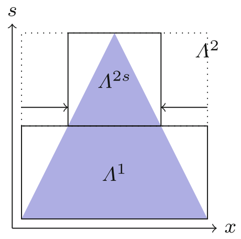

Fig. 8 Bounding boxes fully within an image form a *triangle* in $x - s$ parameter space, while the interval set description leads to *rectangles*. The shrinkage step adjusts interval limits of Λ^2 to describe the *smallest enclosing rectangle* Λ^{2s}; Λ^1 remains the same. This suppresses bounding boxes partially outside the image

5.2 Including Set Shrinkage

Next we describe a parameterisation that is largely equal to the one described above, but includes a *shrinkage* step. The previous parameterisation allowed bounding boxes that can be partially outside the image. Although these bounding boxes are generally low-scoring (due to fewer supporting features), one may want to eliminate them explicitly.

Representation & Initialisation as in Sect. 5.1.

Constraints & Shrinkage. We apply a shrinkage step to enforce additional constraints on the bounding boxes. Thereby we can ensure that e.g.bounding boxes partially outside the image get eliminated. One constraint is that the box width w must be smaller than the image width W. Moreover, the object center x of larger objects have to be farther away from the image border. This yields the constraints $w < W$ and $w/2 \leq x \leq W - w/2$ (similar for y, h, H) which can also be expressed in terms of ls, lr. The shrinkage step uses these constraints to adjust the interval limits as shown in Fig. 8. The update has to be conservative in order not to lose any bounding box that is fully within the image. For example, the constraint $ls - lr < \log H$ yields new scale upper limit $\min(\overline{ls}, \log W - \underline{lr})$ and aspect-ratio lower limit $\max(\underline{lr}, \underline{ls} - \log H)$. Similarly for other constraints we obtain the shrinkage rules:

$$\underline{x} = \max(\underline{x}, w_{\min}/2) \quad \overline{x} = \min(\overline{x}, W - w_{\min}/2) \tag{12}$$

$$\underline{y} = \max(\underline{y}, h_{\min}/2) \quad \overline{y} = \min(\overline{y}, H - h_{\min}/2) \tag{13}$$

$$\underline{lr} = \max(\underline{lr}, \underline{ls} - \log H) \quad \overline{lr} = \min(\overline{lr}, \log W - \underline{ls}) \tag{14}$$

$$\overline{ls} = \min(\overline{ls}, \log H + \overline{lr}, \log W - \underline{lr}) \tag{15}$$

with $w_{\min} = \exp(\underline{ls} + \underline{lr})$ and $h_{\min} = \exp(\underline{ls} - \overline{lr})$. Finally, sets with interchanged interval limits (e.g., $\underline{x} > \overline{x}$) are empty and get eliminated.

Bounding Box & Set Size. This shrinkage procedure is applied after every set splitting and ensures bounding boxes (partially) outside the image are excluded from the search. Consequently, the union of bounding boxes as computed pre-

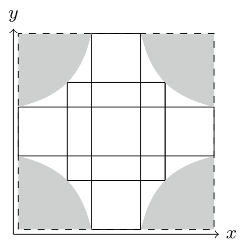

Fig. 9 A set of fixed scale and variable aspect-ratio bounding boxes *solid*. Its union bounding box *dashed* covers pixels *grey* not part of any single box; that might challenge a detector

viously can be clamped to the image domain. The size of a (shrinked) set is computed as before in Sect. 5.1.

Splitting. Equations (12, 13) show that the spatial intervals depend on the size of the bounding box: the larger the bounding box the smaller the interval. As the splitting procedure relies on the interval size relative to the largest bounding box we also account for the shrinkage when choosing the split dimension. For example, the the x-interval size becomes $(\min(\overline{x}, W - w_{\max}/2) - \max(\underline{x}, w_{\max}/2))/w_{\max}$ with $w_{\max} = \exp(\overline{ls} + \overline{lr})$.

5.3 Position, Width, and Height

Representation & Initialisation. The argument for the previous representation was to control the aspect-ratio upon initialisation, but we found that additional constraints need still to be enforced. Moreover, the bounding box union can cover pixels not part of any bounding box which seems inappropriate (Fig. 9). Using scale&aspect-ratio explicitly thus provides no real advantage; we now represent a bounding box by its center (x, y), width w, and height h, and control the aspect-ratio using the shrinkage step. The set description results from extending all four coordinates to intervals

$$\Lambda = [\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}] \times [\underline{w}, \overline{w}] \times [\underline{h}, \overline{h}] \tag{16}$$

and the search space for an $W \times H$ image is

$$\Lambda^0(I) = [0, W] \times [0, H] \times [s_{\min}, W] \times [s_{\min}, H] \tag{17}$$

where we again choose $s_{\min} = 50$ pixels.

Constraints & Shrinkage. We derive the shrinkage rules as before, but need to enforce different constraints. In particular, we have to ensure that $r_{\min} \leq w/h \leq r_{\max}$ and $w/2 \leq x \leq W - w/2$ (similar for y). The constraints $w < W$ and $h < H$ always hold. The shrinkage rule thus becomes

$$\underline{x} = \max(\underline{x}, \underline{w}/2) \qquad \overline{x} = \min(\overline{x}, W - \underline{w}/2) \tag{18}$$

$$\underline{y} = \max(\underline{y}, \underline{h}/2) \qquad \overline{y} = \min(\overline{y}, H - \underline{h}/2) \tag{19}$$

$$\underline{w} = \max(\underline{w}, r_{\min}\underline{h}) \qquad \overline{w} = \min(\overline{w}, r_{\max}\overline{h}) \tag{20}$$

$$\underline{h} = \max(\underline{h}, \underline{w}/r_{\max}) \qquad \overline{h} = \min(\overline{h}, \overline{w}/r_{\min}). \tag{21}$$

Bounding Box & Set Size. The union of bounding boxes is simply $B(\Lambda) = [\underline{x} - \overline{w}/2, \underline{y} - \overline{h}/2, \overline{x} + \overline{w}/2, \overline{y} + \overline{h}/2]$ as the representation stores the maximal width/height. As we apply the shrinkage, we can clamp this bounding box to the image domain. Furthermore, we define the set size as the product of its intervals, i.e.,

$$|\Lambda| = \frac{(\overline{x} - \underline{x})}{\underline{w}} \frac{(\overline{y} - \underline{y})}{\underline{h}} \frac{(\overline{w} - \underline{w})}{\underline{w}} \frac{(\overline{h} - \underline{h})}{\underline{h}}, \tag{22}$$

where all interval sizes are measured relative to the objects size. This ensures scale-invariance while we again normalise w.r.t. to the smallest scale-objects (which dominate the size of a set).

Splitting: Binary&Quadruple Splits. For splitting, we compute the scale-normalised interval sizes (y, h analogue)

$$\frac{\min(W - \overline{w}/2, \overline{x}) - \max(\overline{w}/2, \underline{x})}{\overline{w}} \qquad \frac{\overline{w} - \underline{w}}{\overline{w}} \tag{23}$$

where the normalisation is relative to the largest-scale objects. We also account for the (potential) shrinkage at that scale which favours w over x splits to avoid too fine a localisation of large objects (as motivated in Sect. 5.1). This representation does not couple horizontal and vertical dimensions and we thus experiment to split them both simultaneously. That is, instead of splitting always one interval, we also split pairs (x, y) or (w, h) simultaneously; this results in four subsets rather than two. As always, a pair with the smallest interval size is split.

6 Experiments

This section evaluates the performance of branch&rank. All details of the evaluation, the pre-processing steps like image feature extraction are described in Sect. 6.1. Then, individual components are analysed in the following sections. We analyse in isolation the classifier (Sect. 6.2), the parametrisation (Sect. 6.3), and the task quantisation (Sect. 6.4). Finally the overall performance (Sect. 6.5) and the efficiency (Sect. 6.6) of the resulting branch&rank detector are evaluated.

6.1 Testbed and Features

Testbed. We use the VOC'07 dataset (Everingham et al. 2007) as a testbed for the experiments, it consists of about 10 k images with 20 classes and comes with three pre-defined splits "train", "val", and "test". We compare different configurations of the branch&rank detector on a subset of 10 classes, while the final (test-set) evaluation and comparison to published results is done for all classes. The evaluation scheme is as follows: the parameter C is estimated by training (validating) models on the train (val) data splits. As images are selected randomly during hard negative mining, we always average over three different runs; the variation

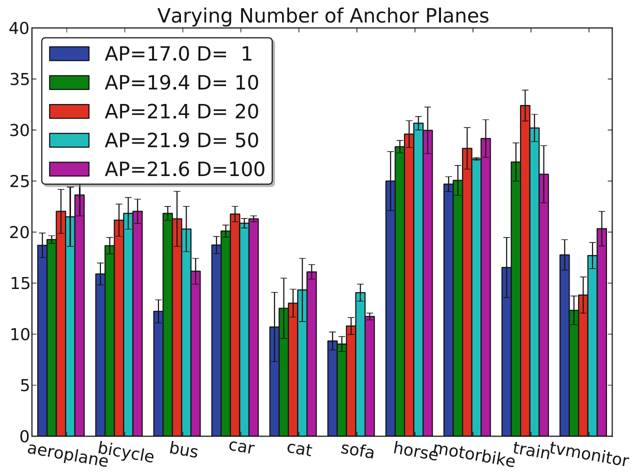

Fig. 10 Detection with anchor plane SVMs. As expected, more anchor planes improves results and outperforms linear SVMs (D = 1). We choose D = 20 for a good trade-off between average precision and computation

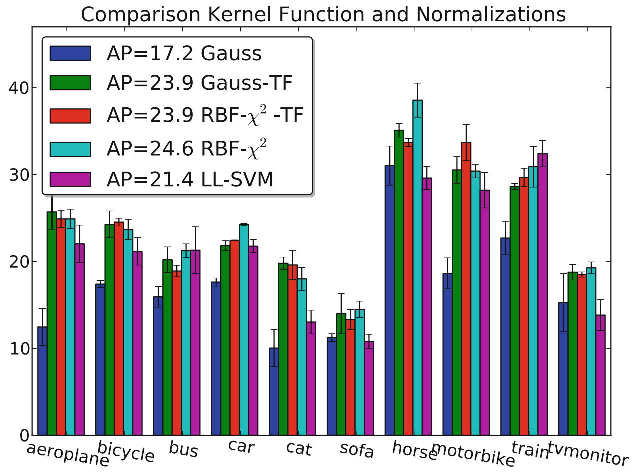

Fig. 11 Detection results with varying kernel functions. The RBF-χ^2 kernel performs best. Taking computation into account, anchor plane SVMs with term frequency (TF) reweighing are competitive: they are a bit worse, but they are an order of magnitude faster. See text for details

among runs is plotted with error bars. Using the best C value, we eventually retrain on the entire 'trainval' split and evaluate on the test data. For training, we ignore images that contain truncated or difficult examples. Moreover, we use only 1,000 negative images during validation. Throughout, we measure detection quality using average precision of VOC'10, unless stated otherwise.

RGB- SIFT- Pyramid- Features. We extract features on a dense grid at multiple scales. We use the code of van de Sande et al. (2010) and select rgb-SIFT descriptors. This typically results in about 15k features that we quantise using a vocabulary of 100 visual words using k-means clustering. Subsequently, we apply a spatial pyramid histogram scheme with 1×1, 2×2, and 4×4 bins that yields a 2100D sub-image descriptor $\phi(\Lambda)$ for a set Λ. In fact, we further apply a normalisation where we compare two versions. One version is to normalise the vector by its l_2-norm. The second to apply term frequency reweighing as common in retrieval (Robertson and Walker 1994). Specifically, we rescale every feature x by $x/(x+a)$ where we found $a = 7$ to work well. This relates to binarisation/max-pooling (Boureau et al. 2010) as each feature saturates at 1, but it does so in a smooth fashion. Results obtained by this normalisation are denoted by a TF suffix.

6.2 Comparison of Kernels

The first experiment validates the quality of different kernel functions. In particular, we elucidate the quality of the anchor plane SVMs compared to common non-linear SVMs. Furthermore, we investigate the difference between the RBF-χ^2 and a standard Gaussian kernel. The latter uses the Euclidean distance to compare feature vectors.In this section and also Sects 6.3 and 6.4 we use the "train" split for training and "val" for performance evaluation of the model.

First of all, Fig. 10 summarises the performance of the anchor plane SVMs using a varying number of planes D; note that the case $D = 1$ is simply a linear SVM. The plot indicates that increasing the dimensionality indeed improves the average precision of the system but levels off at about $D = 20 - 50$. Using locally linear SVMs with D=50 anchor planes yields a mean average precision of 21.9 % as opposed to 17.0 % in case of linear SVMs. As computation time and storage requirements scale linearly with the dimensionality, we prefer fewer dimensions. Therefore, we fix $D = 20$ in all subsequent experiments as this choice provides a good trade-off between speed and quality (i.e., mAP = 21.4 %).

Next we study the performance of a variety of kernel functions. We compare the Gaussian kernel and the RBF-χ^2 kernel, both with l_2 or term-frequency (TF) reweighing. Moreover, we also compare with anchor plane SVMs that apply the TF reweighing. The results are displayed in Fig. 11. We observe that the RBF-χ^2 kernel with l_2 reweighing (RBF-χ^2-TF) yields best performance (mAP = 24.6 %); this finding is in line with results in the literature (Gehler and Nowozin 2009). Moreover, we see that the Gaussian kernel is competitive when using the term frequency reweighing (Gauss-TF yields mAP= 23.9 %), while the Euclidean distance is not appropriate for the raw feature counts (Gauss yields mAP=17.2 %). We thus conclude that a re-weighting scheme such as TF or an adaption of the distance measure, e.g. using χ^2 distance, provides better results.

While the performance decreases a bit, the pay-off from locally-linear SVMs become apparent when looking at the computation and storage requirements. We report numbers for *car* as an exemplary example. The entire evaluation (including training, hard-negative mining, and testing on the

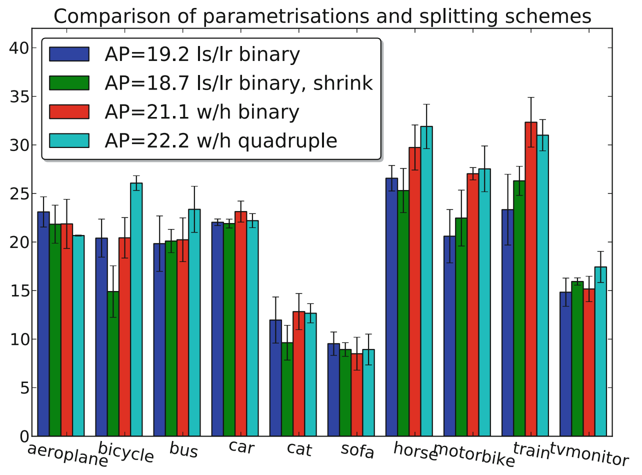

Fig. 12 Search space partitioning schemes. We parametrise objects by their center and scale&aspect-ratio (ls/lr) or width&height (w/h), respectively. Moreover, all but the first variant include a shrinkage step to control the aspect-ratio and to ensure that bounding boxes are fully within the image. Width&height parametrisation yields the best results in particular with quadruple splits

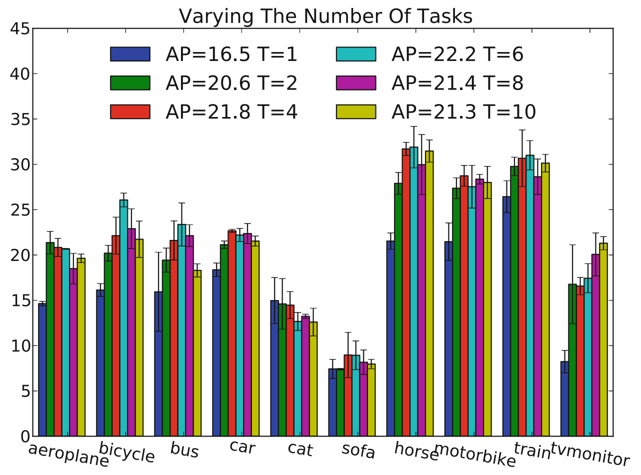

Fig. 13 Multi-Task Ranking: Grouping sets into multiple tasks outperforms the monolithic approach (T = 1) significantly. The mean average precision suggest T = 6 to be a good choice

validation set) takes roughly 25 min for an anchor plane ($D = 20$) SVM, compared to 250 min for the RBF-χ^2 classifier. This is one order of magnitude less runtime. In terms of storage, the models are 10MB and 66MB large, respectively. Hence, measuring performance not only in terms of "average precision" makes anchor plane SVMs competitive.

In conclusion, the RBF-χ^2 kernel yields best accuracy (24.6 %), while anchor plane SVMs have strong computational advantages with only minor decrease in accuracy (21.4 %). In the sequel, we therefore compare various detector configurations (i.e., different search space partitioning schemes, increasingly many tasks) using the much faster anchor plane SVMs. However, the final evaluation and comparison to published results is done using the more accurate RBF-χ^2 kernel.

6.3 Search Space Partitioning Schemes

Section 5 described several search space partitioning schemes that we now compare in terms of AP using anchor plane SVMs. Recall, there are two different parametrisation based on width&height (w/h) and scale&aspect-ratio (ls/lr), respectively. For the latter, we either explicitly suppress bounding-boxes (partially) outside the image (using a shrinkage step) or assume that these boxes generally have a low score (due to the lack of features). The width&height parametrisation always includes the shrinkage step (to control an object's aspect-ratio), but we experiment with binary or quadruple splits. The results are reported in Fig. 12 and indicate that the width&height parametrisation is slightly better than using scale&aspect-ratio. The reason might be that the sub-image that covers all bounding boxes of a set is usually more compact (due to the problems illustrated in Fig. 9). At first sight, splitting into four subsets seems slightly superior, but this is mainly due to the result on *bicycle*. Looking at the other classes, the two variants perform similar. As we think it is adequate to simultaneously partition horizontal (i.e., x, w) and vertical (y, h) axis intervals, we subsequently work with the quadruple split approach. Moreover, this splitting gave the best accuracy (22.2 %).

6.4 Multi-task Improvements

This experiment investigates the benefit of our multi-task framework as proposed in Sect. 4.1. The supposition is that the visual appearance within the union of bounding boxes is not sufficient to properly rank a hypothesis set. Therefore, we argue that grouping sets into tasks should increase the detection quality. The effect of using $T \in \{1, 2, 4, 6, 8, 10\}$ tasks is reported in Fig. 13. Our supposition holds that a holistic detector without any task decomposition (T = 1) performs worse (16.5 %). The best performance is attained with T = 6 tasks (22.2 %) while too fine a quantisation (i.e., T = 10) deteriorates slightly (21.3 %). This is also expected as the number of (per-task) training examples decreases. In summary, the transition from the initial image classification to the eventual object recognition task is well covered with T = 6 tasks. This yields a smooth transition and increases performance w.r.t.the holistic T = 1 ranker by about 6 %. We thus conclude that the decomposition into different tasks is necessary for reasonable performance.

6.5 Performance Evaluation

Finally, we evaluate the detector on the VOC'2007 testset and use the 'trainval' split for training. The detector distinguishes

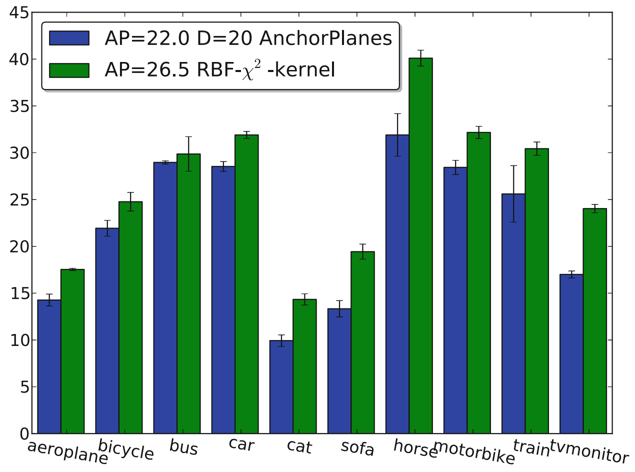

Fig. 14 Comparison of RBF-χ^2-kernel with anchor plane SVMs on the testset (trained with trainval subset). The true non-linearity of the former yields better average precision results. The latter is 4.5 % worse, but it is about 10 times faster

six tasks and uses the width&height parametrisation with quadruple splits. Figure 14 reports the results of the anchor plane SVM with 20 planes and the non-linear RBF-χ^2-SVM. The RBF-χ^2 SVM clearly outperforms anchor plane SVMs in terms of average precision; recall that anchor plane SVMs are an order of magnitude faster.

Next we compare the performance of branch&rank with a state-of-the-art detector (Felzenszwalb et al. 2008) (dt) as well as the best (per category) results reported in the challenge (Everingham et al. 2007) (v7).[4] In order to compare results with the published results (Felzenszwalb et al. 2008; Everingham et al. 2007), we use the original VOC'07 AP measure which has a sampling artefact[5]; the previous results were reported using the newer VOC'10 AP measure that resolved this issue. Table 2 shows that our scores are sometimes higher (horse, sofa), lower (e.g.bicyc, bus, car), or in between (e.g.aeroplane, cat, motorbike, train). This is the ranking we have expected using only one single un-tuned image descriptor. Looking at the results of other contestants (not only the per-class winners) we found that our scores are in a similar range and conclude that for certain categories (e.g., bottle, person) a combination of multiple diverse image descriptors is vital to achieve even higher accuracy. In fact, (Vedaldi et al. 2009) convincingly demonstrated that combining multiple complementary features significantly improves detection quality. A comparison to (Vedaldi et al. 2009) is out of the scope of this paper as we use only one single feature. Our aim is to improve detection efficiency and the algorithmic properties can well be demonstrated with a single image descriptor. The next section will demonstrates the efficiency of branch&rank.

6.6 Efficiency: often less than 100 classifier calls

We measure the efficiency of the proposed detector in terms of number of classifier evaluations as opposed to runtime in seconds. The latter is certainly of practical interest and we partially addressed this topic with the anchor plane SVM experiments. However, this work focuses on efficiency where we aim for algorithms that scale sub-linearly in the number of hypotheses (c.f. Fig. 1). In Fig. 15 the number of iterations (of Algorithm 1) till a detection is plotted (as a function of the detection score) and the average number of iterations (at precision equals recall) is reported in the legends. Our system detects most objects quickly: the averaged number of iteration at precision=recall is about 30 iterations (that is $4 \times 30=120$ classifier calls) while the high scoring detection are found even quicker. This finding reinforces our conjecture from Sect. 3.5: a better ranking improves detection efficiency. For comparison, ESS (Lampert et al. 2009) reports on the order of 10 k iterations, while Vedaldi et al. (2009) score 100 boxes in the last non-linear stage (thus on top of the early cascade evaluations). In conclusion, our method is efficient making it possible to solely use non-linear SVMs. Runtime is affected by the classifier evaluation and the number of times it is called. On the class *car* detection with the non-linear χ^2 kernel SVMs took around 8.5 s per image, the local linear classifier runs at 0.5 s per image.

7 Conclusion

In short, branch&rank (Lehmann et al. 2011a) generalises the idea of branch&bound (Lampert et al. 2009; Lehmann et al. 2011b): ranking improves efficiency and thereby enables the use of arbitrary classifiers, including non-linear SVMs with RBF-χ^2 kernels. This is a crucial advance in efficient object detection since strong classifiers are beneficial to properly model the object intra-class variations. Let us recapitulate, the efficiency of our method results from leveraging the branching step of branch&bound, but superseding the bounding by a *ranking* step. This relieves the former limitations (availability of a tight bounding function) and allows for arbitrary ranking functions. The system is trained in a structured SVM setting while a multi-task formulation has proven effective: it properly handles image classification, object recognition, and in-between task arising throughout the search procedure. The experiments show that branch&rank localises objects using often less than 100 classifier calls. This efficiency enables costly and thus strong classifiers.

[4] We cannot directly compare to Lampert (2010) as they evaluate using recall-overlap rather than average precision

[5] Due to measuring the performance at a discrete set of recall values, an AP of $1/11 \approx 9\,\%$ is obtained if the best-scoring detection is correct *even if it is the only one*.

Table 2 Average precision (AP) on the VOC 2007 testset; detectors are trained with the 'trainval' and evaluated on the 'test' datasplit

		Aerop	Bicyc	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Dtable	Dog	Horse	Mbike	Person	Plant	Sheep	Sofa	Train	Tv
AP10	Anchor Plane	14.3	21.9	1.7	3.4	1.3	29.0	28.5	9.9	1.2	7.1	3.0	7.3	31.9	28.4	5.7	0.5	7.9	13.3	25.6	17.0
	RBF-χ^2	17.5	24.8	3.7	6.6	2.1	29.9	31.9	14.3	1.8	9.5	5.4	12.6	40.1	32.2	7.3	2.1	11.4	19.4	30.4	24.0
AP07	RBF-χ^2	20.6	26.9	9.5	11.4	8.1	32.0	32.5	17.3	6.8	12.9	10.5	15.9	41.5	33.8	13.4	8.4	14.5	23.2	32.1	26.5
	v7	18.0	41.1	9.2	9.8	24.9	34.9	39.6	11.0	15.5	16.5	11.0	6.2	30.1	33.7	26.7	14.0	14.1	15.6	20.6	33.6
	dt	26.2	40.9	9.8	9.4	21.4	39.3	43.2	24.0	12.8	14.0	9.8	16.2	33.5	37.5	22.1	12.0	17.5	14.7	33.4	28.9

The top two rows show AP scores using the new (2010) AP implementation; the non-linear RBF-χ^2-SVMs performs about 3.5 % better than the faster anchor plane SVMs. The bottom three rows use the old 2007 AP score to allow for comparing branch&rank with a state-of-the-art detector (Felzenszwalb et al. 2008) (dt) as well as the best (per category) results reported in the challenge (Everingham et al. 2007) (v7). (Note that rows 2&3 differ only in terms of AP score implementation.)

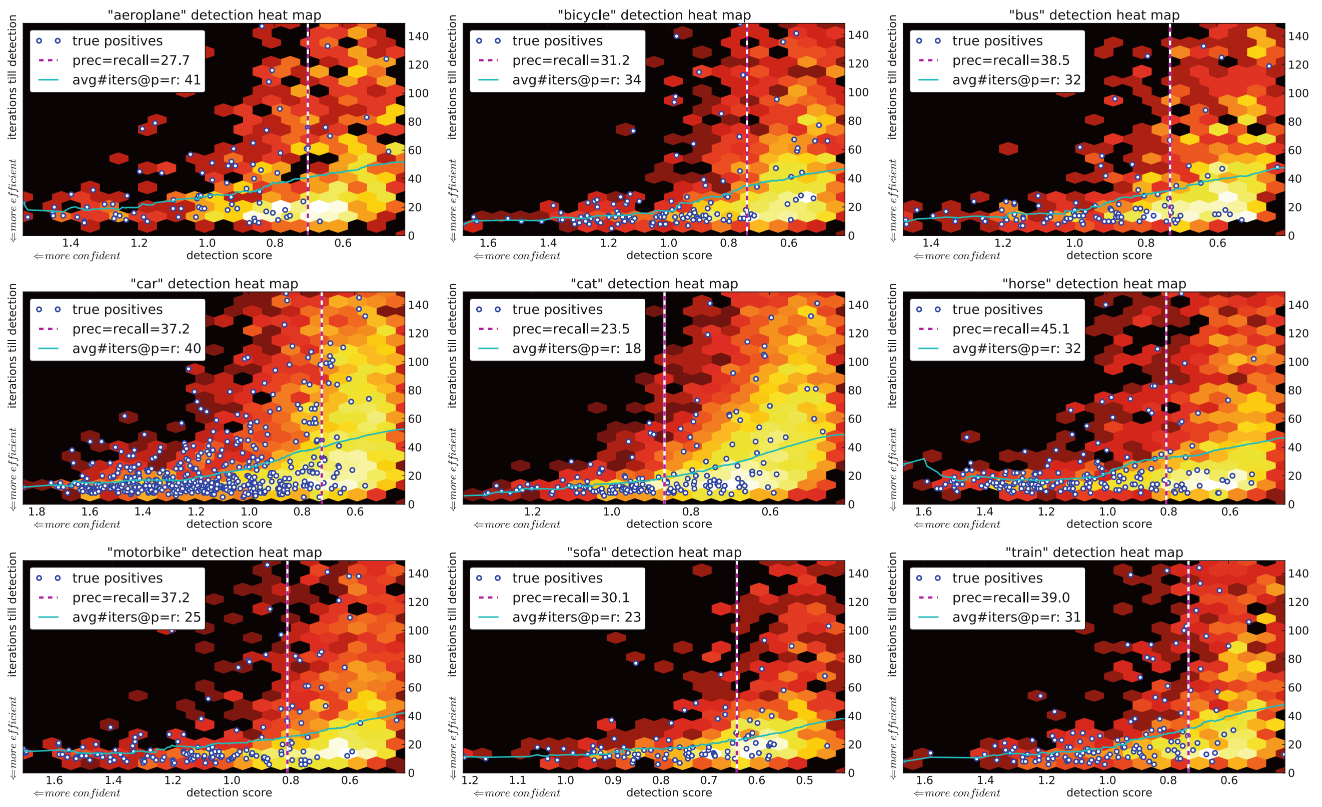

Fig. 15 Detection efficiency: iterations till detection versus object detection score. The heat map represents the joint (score-iterations) density from all detections reported on the VOC'07 testset; bright indicates high density. The *blue circles* denotes the true positive detections only and the *cyan line* shows the cumulative average number of iterations. Branch&rank uses on average less than 60 iterations, about 30 at precision=recall, and sometimes only about 10 (Color figure online)

A novel aspect of branch&rank is that the notion of sets is already integrated into the training. The ranking function can therefore leverage information of a set which is not available when looking at a single bounding box. This allows to overcome a systematic bias (towards larger sets) of bounding functions (c.f. Lehmann 2011) and thus improves efficiency. As a result, we made detection by non-linear SVMs feasible, without the need for approximations.

Let us emphasise that efficiency is orthogonal to reducing the *cost* of a classifier. Using faster classifiers will eventually reduce the overall runtime as we showed with anchor plane SVMs. While those performed a bit worse, they run an order of magnitude faster. This enables to combine multiple complementary features, which are the source of most empirical progress in image classification and detection (Gehler and Nowozin 2009; Vedaldi et al. 2009). This is subject to future work while we found a single feature sufficient to demonstrate the algorithmic properties of branch&rank.

Multi-task aspects play a vital role in branch&rank: hypothesis sets throughout the search process correspond to image classification, object categorisation, and in-between tasks. We captured this phenomenon by a *task mapping* that groups related sets; each task is scored with a dedicated function that still targets a global ranking. This concept is versatile and our grouping (based on the set size) only scratched the surface of what is possible. For example, (Park et al. 2010; Zhang et al. 2011b) group by scale (and aspect-ratio) to better cope with small, low-resolution objects. Our task mapping describes such grouping in a formal, yet simple and general manner.

In the future, we plan to better leverage the power of this task mapping. The flexibility of branch&rank in fact allows to use different appearance descriptors for different tasks, and to sample features on demand. The next step is thus to take advantage of findings from the image classification and object categorisation community. We envision to improve the detection results by tailoring the ranking function for each task separately. Furthermore, it would also be interesting to automatically learn a task mapping from training data.

Another upcoming research challenge lies in developing a better understanding of hypothesis sets and how to partition them. Our current bi/quad-section scheme is simple yet effective, but it is not directly applicable to e.g., multi-class scenarios. However, we anticipate that the proposed multi-task approach extends to multi-class branching e.g., (Yeh et al. 2009). Designing an appropriate splitting scheme that interleaves spatial and class branching is a promising endeavour: extending branch&rank will provide a principled and efficient true multi-class detector.

References

Alexe, B., Deselaers, T., & Ferrari, V. (2010). What is an object?. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Bileschi, S., & Wolf, L. (2005). A unified system for object detection, texture recognition and context analysis based on the standard model feature set. In *Proceedings of the British Machine Vision Conference*.

Blaschko, M. B. (2011). Branch and bound strategies for non-maximal suppression in object detection. In *Energy Minimazation Methods in Computer Vision and Pattern Recognition*.

Blaschko, M. B., & Lampert, C. H. (2008). Learning to localize objects with structured output regression. In *European Conference on Computer Vision*.

Blaschko, M. B., & Lampert, C. H. (2009). Object localization with global and local context kernels. In *Proceedings of the British Machine Vision Conference*.

Blaschko, M. B., Vedaldi, A., & Zisserman, A. (2010). Simultaneous object detection and ranking with weak supervision. In *Advances in Neural Information Processing Systems*.

Bottou, L., & Bousquet, O. (2008). The tradeoffs of large scale learning. In *Proceedings of the Advances in Neural Information Processing Systems*.

Boureau, Y. L., Ponce, J., & LeCun, Y. (2010). A theoretical analysis of feature pooling in vision algorithms. In *International Conference on Machine Learning*.

Breuel, T. M. (2002). A comparison of search strategies for geometric branch and bound algorithms. In *European Conference on Computer Vision*.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., et al. (2005). Learning to rank using gradient descent. In *International Conference on Machine Learning*.

Carreira, J., & Sminchisescu, C. (2010). Constrained parametric min-cuts for automatic object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Chapelle, O., & Keerthi, S. S. (2009). Efficient algorithms for ranking with svms. *Information Retrieval Journal*, *13*(3), 201–215.

Chum, O., & Zisserman, A. (2007). An exemplar model for learning object classes. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Desai, C., Ramanan, D., & Fowlkes, C. (2009). Discriminative model for multi-class object layout. In *International Conference on Computer Vision*.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2007). *The PASCAL Visual Object Classes, Challenge 2007 (VOC2007) Results*.

Felzenszwalb, P., McAllester, D., & Ramanan, D. (2008). A discriminatively trained, multiscale, deformable part model. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Felzenszwalb, P., Girshick, R., & McAllester, D. (2010). Cascade object detection with deformable part models. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Gall, J., & Lempitsky, V. (2009). Class-specific hough forests for object detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Gangaputra, S., & Geman, D. (2006). A design principle for coarse-to-fine classification. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Gehler, P. V., & Nowozin, S. (2009). On feature combination for multi-class object classification. In *International Conference on Computer Vision*.

Griffin, G., Holub, A., & Perona, P. (2007). *Caltech-256 object category dataset*. Pasadena: California Institute of Technology. (Tech. Rep. 7694).

Harzallah, H., Jurie, F., & Schmid, C. (2009). Combining efficient object localization and image classification. In *International Conference on Computer Vision*.

Itti, L., Koch, C., & Niebur, E. (1998). A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*, 1254–1259.

Keysers, D., Deselaers, T., & Breuel, T. M. (2007). Geometric matching for patch-based object detection. *Electronic Letters on Computer Vision and Image Analysis*, *6*(1), 44–54.

Ladický, L., & Torr, P. H. (2011). Locally linear support vector machines. In *International Conference on Machine Learning*.

Lampert, C. H. (2010). An efficient divide-and-conquer cascade for nonlinear object detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Lampert, C. H., & Blaschko, M. B. (2009). Structured prediction by joint kernel support estimation. *Machine Learning*, *77*, 249–269.

Lampert, C. H., Blaschko, M. B., & Hofmann, T. (2009). Efficient sub-window search: A branch and bound framework for object localization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *99*(1), 2129–2142.

Lazebnik, S., Schmid, C., & Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (Vol. 2, pp. 2169–2178). Los Alamitos, CA, USA.

Lehmann, A., Leibe, B., & Van Gool, L. (2009). Feature-centric efficient subwindow search. In *International Conference on Computer Vision*.

Lehmann, A., Gehler, P., & Gool, L. V. (2011a). Branch&Rank: Efficient non-linear object detection. In *Proceedings of the British Machine Vision Conference*. Dundee, UK.

Lehmann, A., Leibe, B., & Van Gool, L. (2011b). Fast prism: Branch and bound hough transform for object class detection. *International Journal of Computer Vision*, *94*(2), 175–197.

Lehmann, A.D. (2011). Efficient object detection. PhD thesis, Eidgenössische Technische Hochschule ETH Zurich. doi:10.3929/ethz-a-006706798. (Diss. Nr. 19868, Hartung-Gorre Verlag, Selected Readings in Vision and Graphics Vol. 69)

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

Murphy, K., Torralba, A., & Freeman, W. T. (2003). Using the forest to see the trees: A graphical model relating features, objects and scenes. In *Advances in Neural Information Processing Systems*. MIT Press.

Oliva, A., & Torralba, A. (2001). Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, *42*(3), 145–175.

Park, D., Ramanan, D., & Fowlkes, C. (2010). Multiresolution models for object detection. In *European Conference on Computer Vision*.

Pedersoli, M., Gonzalez, J., Bagdanov, A., & Villanueva, J. J. (2010). Recursive coarse-to-fine localization for fast object detection. In *European Conference on Computer Vision*.

Prisacariu, V., & Reid, I. (2009). Fasthog—A real-time gpu implementation of hog. Oxford: Department of Engineering Science, Oxford University. (Tech. Rep. 2310/09).

Razavi, N., Gall, J., & VanGool, L. (2011). Scalable multi-class object detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Robertson, S. E., & Walker, S. (1994). Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *ACM SIGIR Conference on Research and Development in Information Retrieval*.

Torralba, A. (2003). Contextual priming for object detection. *International Journal of Computer Vision*, *53*(2), 169–191.

Torralba, A., Murphy, K. P., & Freeman, W. T. (2010). Using the forest to see the trees: Exploiting context for visual object detection and localization. *Commun ACM*, *53*(3), 107–114.

Tsochantaridis, I., Hofmann, T., Joachims, T., & Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, *6*, 1453–1484.

van de Sande, K. E. A., Gevers, T., & Snoek, C. G. M. (2010). Evaluating color descriptors for object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(9), 1582–1596.

Vedaldi, A., Gulshan, V., Varma, M., & Zisserman, A. (2009). Multiple kernels for object detection. In *International Conference on Computer Vision*.

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

Wei, Y., & Tao, L. (2010). Efficient histogram-based sliding-window. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Weiss, D., Sapp, B., & Taskar, B. (2010). Sidestepping intractable inferece with structured ensemble cascades. In *Advances in Neural Information Processing Systems*.

Wojek, C., Dorkó, G., Schulz, A., & Schiele, B. (2008). Sliding-windows for rapid object class localization: A parallel technique. In *DAGM-Symposium* (pp. 71–81).

Wolf, L., & Bileschi, S. (2006). A critical view of context. *International Journal of Computer Vision*, *69*(2), 251–261.

Yeh, T., Lee, J. J., & Trevor Darrell, T. (2009). Fast concurrent object localization and recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Zhang, Z., Ladick, L., Torr, P. H., & Saffari, A. (2011a). Learning anchor planes for classification. In *Advances in Neural Information Processing Systems*.

Zhang, Z., Warrell, J., & Torr, P. H. S. (2011b). Proposal generation for object detection using cascaded ranking svms. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.