Branch&Rank: Non-Linear Object Detection

Alain D. Lehmann[1]
lehmann@vision.ee.ethz.ch

Peter V. Gehler[1,2]
pgehler@mpi-inf.mpg.de

Luc Van Gool[1,3]
vangool@vision.ee.ethz.ch

[1]Computer Vision Laboratory
ETH Zurich, Switzerland

[2]Computer Vision & Multim. Comp. Lab
MPI for Informatics, Germany

[3]ESAT-PSI/IBBT
KU Leuven, Belgium

Abstract

Branch&rank is an object detection scheme that overcomes the inherent limitation of branch&bound: this method works with *arbitrary* (classifier) functions whereas *tight* bounds exist only for simple functions. Objects are usually detected with less than 100 classifier evaluation, which paves the way for using strong (and thus costly) classifiers: We utilize non-linear SVMs with RBF-χ^2 kernels without a cascade-like approximation.

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 sub-dividing 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 too simple function classes). The grouping makes the system effective: it separates image classification from object recognition, yet combines them in a single, structured SVM formulation. A novel aspect of branch&rank is that a better ranking function is expected to decrease the number of classifier calls during detection. We demonstrate the algorithmic properties using the VOC'07 dataset.

1 Introduction

Object class detection in images is challenging because of two problems. First, objects 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. Much progress has been made lately, manifesting itself in increasing evaluation scores of the VOC benchmark [8]. This benchmark exclusively measures detection accuracy although computational aspects are also relevant in practice. This paper puts an emphasis on *detection efficiency* and we pose the following two demands: detectors must cope with the appearance variations and must handle the large search space (of possible object positions) efficiently.

The appearance variations are best captured by using strong classifiers and by combining different image descriptors. Non-linear SVMs are consistently found to be well suited for this task [13, 26]. Unfortunately, such classifiers are expensive to evaluate which makes it challenging to master the large search space: exhaustive 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 [26, 27] or

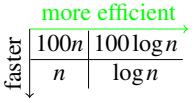


Figure 1: Fast *vs.* efficient search within a total of n possible object candidates.

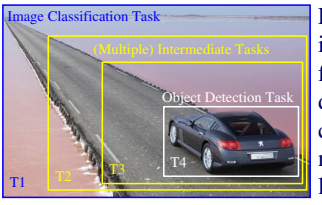


Figure 2: We decompose object detection into different tasks: our approach blends from image classification (T1) to object detection (T4). This allows for object localization with runtime sub-linear in the number of candidate regions, often using less than 100 classifier evaluations.

(B) reducing the number of classifier calls [17, 19]. Let us stress this point: we distinguish between the cost of the classifier and the number of times it is executed. These two factors are orthogonal and their product yields the total runtime (*c.f.* Fig. 1). We next discuss the two options separately, but note that they can also be combined (*e.g.*, [16, 28]). This paper focuses primarily on (B) and, this said, outperforming the best published result in [8] is not the intention of this paper.

Cascade classifiers [10, 26, 27] are prominent examples to (A) and reduce the classifier cost. They reject many hypothesis 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. We consider these approaches to be fast but not efficient as they do not *scale* well to *e.g.* multi-class scenarios (*c.f.* Fig. 1). Branch&bound methods [17, 19] fall into the category (B). They reduce the number of calls by avoiding exhaustive search. This is possible by operating on *sets* of hypotheses. The detector adaptively partitions the search space and focuses on the most promising set. This best-first search allows for impressive runtime given a *tight* bound on the classifier function. Tight bounds are however a severe limitation as they are often unavailable expect for simple function classes *e.g.* linear SVMs.[1]

This paper supersedes the notion of bounding and thereby allows for using 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 efficient and detects objects with often less then 100 ranking operations. Although the ranking paradigm can be applied 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 [13, 18, 26], but are generally perceived as being too slow to be directly applicable; we here 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 (*c.f.* Fig. 2).

In summary, the concept of ranking has the following benefits: (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 faster* 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, costly classifiers (like non-linear SVMs) without the need for cascade-like approximations. In the sequel, we discuss related work (Sec. 1.1), introduce branch&rank (Sec. 2), extend it to multiple tasks (Sec. 3), followed by experiments (Sec. 4).

[1] Even a linear SVM on normalized histograms does not give rise to an efficient branch&bound algorithm.

1.1 Related Work

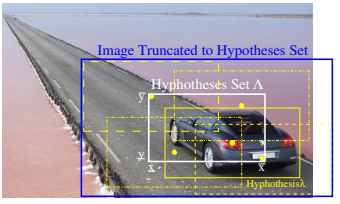
Branch&bound for bounding box detection [17] pioneered the field of efficient object detection where the number of classifier calls scales sub-linearly in the search space size. Its efficiency depends on the availability of a tight bounding function, that, unfortunately, is available for only very small function classes. The obvious bounds for non-linear SVMs are simply not sufficiently tight to be of practical value [26]. 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 [19]. But again, this was only shown for simple function classes. Moreover, [19] gives up on global optimality guarantees, a much appreciated property of branch&bound [17]. We will elaborate on optimality and discuss the properties of our method later in Sec. 2.2.

Coarse-to-fine detectors [12, 22] also operate on sets. However, they start with a uniform partitioning at the coarse level that still scales linearly (similar to sliding-window); only the finer levels partition subsets in an adaptive way. [12] use a cost-to-power criterion to learn how to partition a given set, while we always bisect sets along the dimension with largest extent. They examine hypothesis sets in a breadth-first order and *prune non-promising sets*. In contrast, we follow best-first strategy that *prioritises promising sets*; we never prune. In fact, the concept of pruning is closely related to the idea of cascade detectors.

Cascades have been used with much success to reduce the cost of classifiers [10, 26, 27]. They use simple criteria to reject many hypotheses and thereby reduce the number of strong classifiers evaluations. However, they still process *every* bounding box exhaustively. In other words cascades are fast but not efficient. Although the cascading in [10] does not examine every possible *part configuration*, the root part (which is reported as detection) is evaluated exhaustively as in sliding-window.

Furthermore, cascading and adaptive sub-division are two orthogonal techniques [16, 28]; we plan to combine them with our ranking technique for further speed improvements. Structured ensemble cascades [28] leverage the coarse-to-fine approach and only subdivide hypotheses sets that have not yet been filtered. Again, they filter while we prioritise sets. Moreover, they work on pose estimation and focus on localising part configuration; their images are "largely focused on a single actor" and thus avoid the object localisation that we tackle in this work. [16] 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.

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, [6] generates class-specific proposals based on discriminative visual words while [1] yields class-independent object positions based on various low-level saliency measures. In object segmentation similar techniques are applied with much success [5]. Such two-step proposal-verification schemes can be quite effective but lack of a joint objective function. Thus they are difficult to optimize and the influence of each single part to the entire system is not trivial to measure. Another work in this direction is object priming [23] where also potential object positions are predicted. This method makes use of global image features (such as gist) but does not aim to accelerate the search process but to prune false positives. Similarly [3, 15] use image classifiers in order to improve object detection scores. However, none of those works consider in-between tasks as we do.



Algorithm 1 Detector(ranking function f, n×m image I)

Queue.enqueue $(\infty, [0,n]\times[0,m]\times[s_{\min},s_{\max}]\times[r_{\min},r_{\max}])$
while True **do**
  $score, \Lambda$ = Queue.dequeue_best()
  if $|\Lambda| \leq \varepsilon$ **return** Λ
  else for Λ_i **in** split(Λ) **do** Queue.enqueue$(f(\Lambda_i), \Lambda_i)$

Figure 3: A hypotheses set (white) with 5 of its elements (yellow), *i.e.*, 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. (Aspect-ratio/scale are fixed to avoid cluttering the figure)

2 Branch and Rank

2.1 Best-first Search

Our test-time search algorithm build on the same adaptive partitioning as branch&bound [17, 19], and we will briefly review its ingredients. The method operates on sets of hypotheses and eventually converges to a single bounding box. While there are several possibilities, we parametrise a bounding box by its center (x,y), scale s, and aspect-ratio r. Consequently, hypotheses sets $\Lambda = \left(I, \underline{x}, \overline{x}, \underline{y}, \overline{y}, \underline{s}, \overline{s}, \underline{r}, \overline{r}\right)$ comprise all bounding boxes in image I with center $(x,y) \in [\underline{x},\overline{x}]\times[\underline{y},\overline{y}]$, and scale/aspect-ratio in the intervals $[\underline{s},\overline{s}]$ and $[\underline{r},\overline{r}]$, respectively (see Fig. 3). The set of all possible bounding boxes is then recursively subdivided to find the best one. Alg. 1 summarises the adaptive, best-first strategy that is governed by a priority queue. Every queued element represents a set of hypotheses that is ordered according to the score of a *ranking function*. The algorithm proceeds as follows. Initially, the whole search space is entered into the queue as a single element. At each step the highest ranking element in the priority queue is split in two smaller sets which are subsequently scored and inserted into the priority queue. In case the highest ranking hypotheses 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. For more details we refer the reader to [19] whose scheme we adopted.

2.2 Ranking Condition

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

$$f(\Lambda^+) > f(\Lambda) \qquad \forall \Lambda^+ \in \mathbb{L}^+, \quad \forall \Lambda \in \mathbb{L} \setminus \mathbb{L}^+, \tag{1}$$

where the set Λ^+ contains at least one object and Λ contains no object (or only with partial bounding box overlap). Learning such a function is the topic of the next Section while we now discuss the method's underlying assumption, convergence, and optimality properties.

Assumption. The successful operation of branch&rank depends on one assumption: it must be possible to decide whether a set (*i.e.*, a certain sub-image) *contains* an object or not. This is exactly the problem that image classification is addressing with much success [8]. This supports our method, but it also suggest 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 tight bound requirement.

Convergence. Imagine a function f^{oracle} that meets the condition (1) for any observable image. In that case, Alg. 1 will first examine all sets that do contain an object and the size

of these sets decreases exponentially fast (as we always split them into two halves). That implies that objects of interest are detected in logarithmic time. Of course, this cannot be expected in practice, but it suggests: the better a ranking function, the better *and faster* the detector.

On optimality. We follow [9] 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 prediction problem.[2] For branch&bound (BB) as in [17] the test-time error is zero, but the approximation error is high due to the limited function classes for which tight bound 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 BB because we can use richer function classes, *e.g.* non-linear SVMs. The recent research on object detection (*e.g.* [26]) indicates that the decrease of the approximation error outweights 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.

2.3 Structured SVM Formulation

Several ranking problems have been studied in the machine learning literature and we adopt structured SVMs using margin-rescaling [24] to train the ranking function. This section gives a high-level presentation of the training problem, while the Sec. 3 eventually specifies the classifier function and image descriptor. The ranking condition from the last section results in the optimisation problem

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

with slack variables ξ_j for every positively annotated example $\{\Lambda_j^+\}_{j=1}^n$ and regularisation parameter C, that trades data fit with model complexity. 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* (as opposed to specific instance) detection, 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.*, $\Delta(\lambda_1^+,\lambda_2^+) = 0$.[3] Therefore, the loss depends on all annotations Y. We use the VOC loss function as in [7] and extend it to sets of bounding boxes as

$$\Delta(\Lambda) := \Delta(\Lambda_j^+,\Lambda) = 1 - \max_{\substack{\lambda_i\in Y \\ \lambda\in\Lambda}} \frac{area(B(\lambda)\cap B(\lambda_i))}{area(B(\lambda)\cup B(\lambda_i))} \tag{3}$$

with bounding box $B(\lambda)$. This definition exploits the fact that during training the first argument is always a positive example. This yields the "$1-$" term which would otherwise be a "max" term similar to the second one. This loss is small if one instance (of the set Λ) has high overlap with a ground truth object (in particular $\Delta(\Lambda^+) = 0$) and is 1 if there is no instance with any overlap.

[2] There is also an optimization error if we solve the learning problem approximately. Using convex formulations this can be neglected.

[3] we use λ for single objects, Λ for sets and $\mathbb{L}$ for sets of sets; a $^+$ indicates that it contains at least one positive.

3 Multi-Task Ranking Problem

From Classification to Detection. The previous section's formulation suggests *one* ranking function for all possible sets of bounding boxes. Let us consider the two extremes of such sets. 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 hypotheses 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 suggest 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 [21], while the latter could make use of object specific features [7, 20, 21]. Moreover, grouping related examples into tasks reduces the intra-task variability which simplifies the learning problem.

Task Mapping. We capture the notion of tasks by a mapping $q(\Lambda) \mapsto \{1,2,\dots T\}$ which assigns a task ID to any hypotheses set. This can be achieved *without* looking at the actual appearance, but by simply analysing the hypotheses set itself. In other words, the set provides valuable *domain knowledge*. The above described difference of classification and recognition tasks lies in the *size* of a set. We define it as the product of interval sizes as

$$|\Lambda| = \frac{\overline{x}-\underline{x}}{\overline{s}} \times \frac{\overline{y}-\underline{y}}{\overline{s}} \times (\log\overline{s} - \log\underline{s}) \times (\log\overline{r} - \log\underline{r}) \tag{4}$$

with a scale normalisation for the spatial intervals (*c.f.* Sec. 2.1). The normalisation measures the spatial interval relative to the actual object size; the logarithm reflects the multiplicative nature of scale and aspect ratio. Our task mapping q discretizes $\log(|\Lambda|)$ uniformly into T different tasks which accounts for the set size's exponential decay (due to the splitting scheme). Let us emphasise that this is one particular choice, while the concept of tasks is more general. Other mappings that *e.g.* group examples by aspect-ratio or that group related classes (in a multi-class setup) could be defined. We set $T = 6$ which blends from image classification to the final object recognition task. Note that both tasks are often dealt with separately [8, 14], while we combine them and complement them with intermediate tasks.

Multi-Task Ranking. This paragraph declares the ranking function which completes the optimisation problem Eq. (2). The multi-task ranking function is of the form

$$f(\Lambda) = \langle w_{q(\Lambda)}, \Phi(\Lambda)\rangle + b_{q(\Lambda)} \tag{5}$$

with per-task weight vectors w_t and bias terms b_t as well as an appearance descriptor $\Phi(\Lambda)$. For simplicity, we use the same descriptor for all task, although we could specifically tailor it to the individual tasks possibly making use of different features. We adopt the commonly used bag-of-words approach [17, 18, 19, 26] that aggregates (local) features and represents them by a histogram. Specifically, $\Phi(\Lambda)$ uses all features that fall within the bounding box $B(\Lambda) := \bigcup_{\lambda\in\Lambda} B(\lambda)$ where the union extends the notion of bounding boxes to sets of hypotheses (*c.f.* Fig. 3, blue box). For large sets, this includes all features extracted from the image as usual in image classification. The optimisation of Eq. (2) eventually benefits from the per-task bias term: it decomposes. Due to a lack of space we simply state the decomposed problem, while the supplementary materials provides a detailed derivation. In fact, we

can learn each pair (w_t, b_t) by solving a smaller optimisation problem

$$\min_{w_t, b_t, \xi \geq 0} \|w_t\|^2 + C \sum_j \xi_j \tag{6a}$$

$$\langle w_t, \Phi(\Lambda_j^+) \rangle + b_t \geq -\xi_j \quad \forall \Lambda_j^+ \in \mathbb{L}^+,\ q(\Lambda_j^+) = t \tag{6b}$$

$$\langle w_t, \Phi(\Lambda) \rangle + b_t \leq -\Delta(\Lambda) \quad \forall \Lambda \in \mathbb{L} \setminus \mathbb{L}^+,\ q(\Lambda) = t. \tag{6c}$$

that uses only examples from one given task t. Let us emphasise that the problem is equivalent to Eq. (2) when using f as in Eq. (5). We optimise the tasks in sequential order (starting with $t = 1$ similar to image classification) using a modified version of SVM^{struct} [24]. This problem can be readily kernelized and we choose to work with the RBF-χ^2-kernel $k(u,v) = \exp\left(-\gamma \sum_l \frac{(u_l - v_l)^2}{u_l + v_l}\right)$ with bandwidth γ.

Delayed Constraint Generation. The constraint set of Eq. (6) is large and includes every possible bounding box. We therefore solve Eq. (6) using delayed constraint generation that, as a side note, can be understood as a formal justification to the commonly used "hard negative mining" (*e.g.*, [7]). This idea was pioneered in [7], while our reformulation deepens the connection to the binary SVM case. We initially generate the positive constraints Eq. (6b) by running Alg. 1 using the *ground truth ranking* $f^{GT}(\Lambda) := \mathbb{I}_{\Lambda \cap Y \neq \emptyset}$.[4] Thereafter, we alternate between optimising Eq. (6) (with the current constraint set) and gathering new examples that violate Eq. (6c). We identify them by running a detector that uses the current estimate of the *loss-augmented score* $f^{LA}(\Lambda) := f(\Lambda) + \Delta(\Lambda)$. Those new examples are so-called *hard-negative* as they are easily confused with positive ones.

4 Experiments

Dataset. The VOC'07 dataset [8] is used as a testbed for the experiments. This dataset consists of 20 classes and a total of about $10k$ images. Our evaluation scheme is as follows: the parameter C is estimated by training (validating) models on the train (val) data splits. Using the best C value, we retrain on the entire trainval split and evaluate on the test data. For training, we use only images with non-truncated objects as well as 100 images containing no object. Performance is measured using the official average precision (AP) score function.

Features. We extract features on a dense grid at multiple scales. We use the code of [25] and select rgb-SIFT descriptors. This typically results in about $15k$ features that we quantise using a vocabulary of 100 visual words (obtained by k-means clustering). The final descriptor $\Phi(\Lambda)$ is a spatial pyramid histogram with 1×1, 2×2, and 4×4 bins. This yields a 2100D feature vector that we normalise by its l_2-norm. We are aware that much more elaborate image features have been proposed and furthermore most empirical progress on image classification and detection can be attributed to the use of multiple complementary features [13, 26]. Although better performance should be expected using multiple features, this single feature with un-optimized codebook is sufficient to demonstrate the algorithmic properties of the proposed method.

Detection Results. We first report the detection results achieved on this dataset. We compare the AP of our branch&rank (b&r) approach with a state-of-the-art detector [11] (dt) as well as the best (per category) results reported in the challenge [8] (v7).[5] Tab. 1 shows that our scores are sometimes higher (dtable,horse), lower (*e.g.* bicyc,boat,bus,car), or

[4] Our method uses exactly the same annotated training data as all other detection approaches.

[5] We cannot compare to [16] as they evaluate using recall-overlap rather than average precision

	aerop	bicyc	bird	boat	bottle	bus	car	cat	chair	cow	dtable	dog	horse	mbike	person	plant	sheep	sofa	train	tv
T1	9.6	12.8	2.3	3.1	1.1	11.4	16.3	11.7	9.1	9.5	5.0	9.3	18.3	15.7	10.0	0.1	2.3	5.5	15.4	10.7
T6	21.8	23.2	2.9	9.8	9.1	20.3	23.0	18.1	9.4	10.8	10.3	9.2	30.0	28.9	11.6	1.5	10.3	13.6	24.9	15.6
baseline	14.8	22.3	0.5	0.8	9.3	30.6	29.4	12.2	9.3	12.8	5.1	13.5	29.6	26.8	13.8	1.8	10.8	13.3	24.7	20.0
b&r	17.3	22.4	0.2	0.6	9.1	27.6	29.3	17.6	3.3	10.8	12.2	13.2	36.8	27.9	14.2	1.7	13.0	15.3	22.5	18.4
dt[11]	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
v7[8]	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
$\#f$	80	70	160	160	160	80	70	60	160	80	80	60	60	70	100	180	80	60	60	100

Table 1: Average precision results on VOC'07. T1,T6: the influence of using 1 or 6 tasks, evaluated on the validation set. The second group shows the final evaluation on trainval/test-split. Branch&rank (b&r) uses 6 tasks and the C value from T6. We score 400 proposals of [1] followed by local refinement (baseline) to estimate the branch&rank's search error. The average number of classifier calls ($\#f$) at prec=recall shows the efficiency of branch&rank.

in between (*e.g.* cat,dog). This is the ranking we have expected using only a single un-tuned image descriptor. The apparently big gap for classes with a score $<10\%$ (*e.g.*, bird, boat, chair, plant) is explained by an artefact of the VOC 2007 devkit [8]:[6] an AP of $1/11 \approx 9\%$ is obtained if the best-scoring detection is correct *even if it is the only one*. 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) one better uses different image descriptors. In fact, [26] convincingly demonstrated that combining multiple complementary features significantly improves detection quality. A comparison to [26] is out of the scope of this paper as we use only one single feature; our aim is to improve detection efficiency. We believe that the algorithmic properties can well be demonstrated with the used single image descriptor. In the remainder, we quantify the search error of our method as well as its efficiency, and we demonstrate the influence of the multi-task setup.

Quantifying the search error. With this experiment we quantify the search error of branch&rank (Alg. 1). The ideal baseline to compare with is sliding windows: scoring every possible bounding box. As this is infeasible, we use the bounding box proposals of [1] as candidate regions to evaluate our classifier on. In accordance with [1] we find that this is indeed a good approximation to exhaustive search and yields high recall rates (right most point of the red lines in Fig. 4). We scored 400 proposals from [1] with the learned function for task T, followed by local refinement (as described in [1]). This baseline uses about 1400 classifier evaluations per image. For branch&rank, we fixed the computational budget to 400 evaluations, *i.e.*, 200 iterations of our algorithm (which invokes two function calls per iteration). Tab. 1 (lines b&r, baseline) reports the results with more details on three categories in Fig. 4. In terms of AP, our method (b&r) with 6 tasks compares well to the baseline. The gain as well as the loss on some classes can be explained by good/bad performance of earlier tasks for which better/worse models have been trained. Improvements can be expected by specifically tune each task by *e.g.* using task-adapted features [21] which we have not done yet. We thus conclude that our system can take advantage of contextual cues (unavailable at the bounding box level) comparable to object priming [23]. We plan to leverage this potential in the future.

Detector Efficiency. We measure the efficiency of branch&rank in number of classifier evaluations as opposed to runtime in seconds (which is about 20*s* per image).The latter would be the measure of choice for cascades which aim for reducing the classifier cost. In this work, the goal is to reduce the number of classifier *calls* as this allows for stronger (and thus often more expensive) classifiers like non-linear SVMs. Fig. 4 plots the number of iterations till a

[6] Details are provided in Section 2 of the supplementary material. Note that the problem is resolved in VOC 2010 [9], but we utilise the 2007 version to allow for comparison with the published results [8, 11].

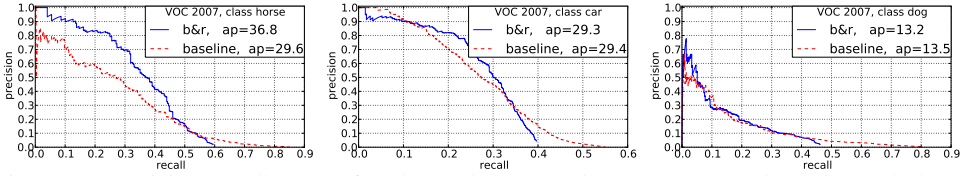


Figure 4: Precision-recall curves for classes *horse,car,dog*. We compare branch&rank (b&r) with 400 proposals of [11] scored with our task T classifier, followed with local refinement as described in [11] (baseline). This baseline provides high recall (right end of red curve) which makes it a good approximation to exhaustive search. b&r compares well to this baseline, which indicates that the search error is small.

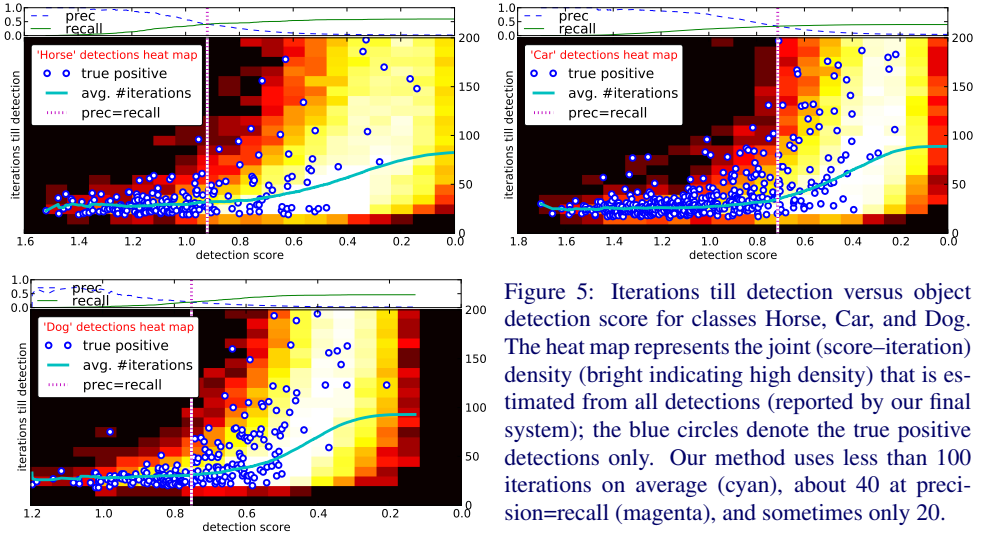


Figure 5: Iterations till detection versus object detection score for classes Horse, Car, and Dog. The heat map represents the joint (score–iteration) density (bright indicating high density) that is estimated from all detections (reported by our final system); the blue circles denote the true positive detections only. Our method uses less than 100 iterations on average (cyan), about 40 at precision=recall (magenta), and sometimes only 20.

detection is reported as a function of the detection score and in Tab. 1 the line $\#f$ shows the average number of calls for all categories. Our system detects most objects rather quickly (in usually less than 50 iterations) especially the high scoring ones but the average number of iterations is also quite low. This finding reinforces our conjecture from Sec. 2.2: the better the ranking, the faster the detector. Higher values of $\#f$ are those with lower scores in Tab. 1. For comparison, ESS [16, 17] reports on the order of $10k$ bound evaluations, while in [26] 100 boxes are scored 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.

Multi-Task comparison. Finally we evaluate the influence of the multi-task ranking proposed in Sec. 3. To this end we train two detectors, one making use of 6 different tasks (T6), and one that is trained without any decomposition (T1). The hyperparameters for both are individually optimized using the validation set. The results are reported in Tab. 1 (lines T1,T6). The multi-task ranking outperforms the holistic one significantly, often with more than 10% difference. The holistic detector T1 handles both the object detection and image classification problem with one ranking function and we conclude that the decomposition into different tasks indeed helps to improve the performance.

5 Conclusion and Future Work

In short, this paper extends the branch&bound method [17] to work with *arbitrary* classifier functions. This is a crucial step towards efficient object detection since the object intra-class variations benefit from the use of strong, non-linear classifiers.

Let us recapitulate, the efficiency of our method results from leveraging the branching step of branch&bound, but superseding the bounding by a set ranking step. That relieves the former limitations (availability of a tight bounding function) and allows for arbitrary (ranking-)classifiers. We train them 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. We envision to improve the detection results by tailoring the ranking function for each task separately. Moreover, one should use multiple and, more importantly, complementary image descriptors. Using diverse descriptors has been found to be the key ingredients for improved detection results *e.g.* using multiple kernels in SVMs [26].

The experiments show that branch&rank localises objects using often less than 100 classifier calls. This efficiency enables strong and thus costly classifiers. Let us emphasise that efficiency is orthogonal to reducing the classifier cost; using cascade approximated classifiers will lead to further speed improvements. Finally, we anticipate that our multi-task approach extends to multi-class branching [29], which would provide a principled and efficient true multi-class detection system.

Acknowledgements

We would like to thank IURO, EC 7FP Strep IURO for funding this project.

References

[1] B. Alexe, T. Deselaers, and V. Ferrari. What is an object? In *CVPR*, 2010.
[2] M. B. Blaschko and C. H. Lampert. Learning to localize objects with structured output regression. In *ECCV*, 2008.
[3] M. B. Blaschko and C. H. Lampert. Object localization with global and local context kernels. In *BMVC*, 2009.
[4] L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In *NIPS*. 2008.
[5] J. Carreira and C. Sminchisescu. Constrained parametric min-cuts for automatic object segmentation. In *CVPR*, 2010.
[6] O. Chum and A. Zisserman. An exemplar model for learning object classes. In *CVPR*, 2007.
[7] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.
[8] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results.
[9] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results. http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html.
[10] P. Felzenszwalb, R. Girshick, and D. McAllester. Cascade object detection with deformable part models. In *CVPR*, 2010.
[11] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *CVPR*, 2008.
[12] S. Gangaputra and D. Geman. A design principle for coarse-to-fine classification. In *CVPR*, 2006.

[13] P. V. Gehler and S. Nowozin. On feature combination for multiclass object classification. In *ICCV*, 2009.
[14] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report 7694, Caltech, 2007.
[15] H. Harzallah, F. Jurie, and C. Schmid. Combining efficient object localization and image classification. In *ICCV*, 2009.
[16] C. H. Lampert. An efficient divide-and-conquer cascade for nonlinear object detection. In *CVPR*, 2010.
[17] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Efficient subwindow search: A branch and bound framework for object localization. *PAMI*, 99(1), July 2009.
[18] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.
[19] A. Lehmann, B. Leibe, and L. Van Gool. Fast prism: Branch and bound hough transform for object class detection. *IJCV*, 94(2):175–197, 2011.
[20] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.
[21] A. Oliva and A. Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *IJCV*, 42(3):145–175, 2001.
[22] M. Pedersoli, J. Gonzàlez, A. Bagdanov, and J. J. Villanueva. Recursive coarse-to-fine localization for fast object detection. In *ECCV*, 2010.
[23] A. Torralba. Contextual priming for object detection. *IJCV*, 53(2):169–191, 2003.
[24] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Large margin methods for structured and interdependent output variables. *JMLR*, 6:1453–1484, 2005.
[25] K. E. A. van de Sande, T. Gevers, and C. G. M. Snoek. Evaluating color descriptors for object and scene recognition. *PAMI*, 32(9):1582–1596, 2010.
[26] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *ICCV*, 2009.
[27] P. A. Viola and M. J. Jones. Robust real-time face detection. *IJCV*, 57(2):137–154, 2004.
[28] D. Weiss, B. Sapp, and B. Taskar. Sidestepping intractable inferece with structured ensemble cascades. In *NIPS*, 2010.
[29] T. Yeh, J. J. Lee, and T. Trevor Darrell. Fast concurrent object localization and recognition. In *CVPR*, 2009.