Branch&Rank: Non-Linear Object Detection Supplementary Material

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

1 Structured SVM decomposition

This section details the decomposition of the structured SVM problem as presented in the main paper (Section 3). The advantage of this decomposition is that the decomposed constraint set facilitates the optimisation. In the sequel, we describe the two steps necessary for the transformation, while a short summary of the main paper's notation is provided in Tab. 1.

Initial Ranking Problem. Recall that the initial optimisation problem for the ranking function f. We follow the margin-rescaling approach of [4] which reads as

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

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 has been defined as

$$\Delta(\Lambda) := \Delta(\Lambda_j^+,\Lambda) = 1 - \max_{\substack{\lambda_i\in Y\\ \lambda\in\Lambda}} \frac{area(B(\lambda)\cap B(\lambda_i))}{area(B(\lambda)\cup B(\lambda_i))} \tag{2}$$

which only depends on one argument. We refer to the main paper for a discussion of this setup.

Step 1: Constraint Decoupling. We exploit the structure of the loss Eq. (2) to decompose the constraint set of Eq. (1). The proposed decomposition builds on an algebraic transformation of 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 $\forall \Lambda_j^+ \in \mathbb{L}^+$, $\forall \Lambda \in \mathbb{L}\backslash\mathbb{L}^+$. The main observation is that each side involves only Λ or Λ_j^+. Moreover, each sides provides a lower (upper) bound for the

Table 1: Notation

λ	a *single* bounding box (parametrisation)
Λ	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*
$B(\lambda)$	the bounding box for parametrisation λ
$\Phi(\Lambda)$	appearance descriptor for hypothesis set Λ
$f : \Lambda \mapsto \mathbb{R}$	ranking function: provide a *priority* for hypothesis set Λ
$q : \Lambda \mapsto \{1,\dots,T\}$	task mapping to distinguish T different tasks.

other. We therefore obtain an equivalent optimisation problem

$$\begin{aligned} &\min_{f,\xi\geq 0,b} \quad \|f\|^2 + C\sum_{j=1}^{n}\xi_j, \\ &\text{s.t.} \begin{cases} f(\Lambda_j^+) + \xi_j \geq b & \forall \Lambda_j^+ \in \mathbb{L}^+ \\ b \geq f(\Lambda) + \Delta(\Lambda) & \forall \Lambda \in \mathbb{L}\backslash\mathbb{L}^+ \end{cases} \end{aligned} \tag{3}$$

that explicitly captures the value b of these upper/lower bounds; note that the value b is part of the optimisation, but it does not occur in the objective function. This step transformed the pairwise constraints from the initial problem into simpler constraints which only depend on one element. This is much in the spirit of binary SVMs, although the constraints for negative examples are not allowed any slack.

Step 2: Multi-Task Ranking Function. Recall that with $q(\Lambda) \mapsto \{1,2,\dots,T\}$ we denoted the mapping of the bounding box set into different task IDs. Depending on the task ID we will use a different linear function $\langle w_{q(\Lambda)}, \cdot\rangle + b_{q(\Lambda)}$ and this is conveniently formalised as a multi-task problem. The resulting whole multi-task ranking function is of the form

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

with per-task weight vectors w_t and bias terms b_t as well as an appearance descriptor $\Phi(\Lambda)$. Substituting this expression into the optimisation problem Eq. (3) yields

$$\min_{w_t,b_t\xi\geq 0,b} \sum_t \|w_t\|^2 + C\sum_j \xi_j \tag{5a}$$

$$\langle w_{q(\Lambda_j^+)}, \Phi(\Lambda_j^+)\rangle + b_{q(\Lambda_j^+)} \geq b - \xi_j \quad \forall \Lambda_j^+ \in \mathbb{L}^+ \tag{5b}$$

$$\langle w_{q(\Lambda)}, \Phi(\Lambda)\rangle + b_{q(\Lambda)} \leq b - \Delta(\Lambda) \quad \forall \Lambda \in \mathbb{L}\setminus\mathbb{L}^+, \tag{5c}$$

which almost decomposes: solely the variable b induces a coupling between the tasks. We further observe that the constraints in fact depend only on the differences $b_t - b$. This difference is translation invariant, *i.e.*, adding a fixed constant to all b_t and b does not affect the solution. In other words, the problem is under-constraint and we are free to add one additional constraint to make the optimal solution unique. We choose the constraint $b = 0$ which eliminates the inter-task coupling. Let us point out that we do not alter the constraint set, we only made a choice of the function class over which we optimise.

Final Optimisation Problem. It remains to group all summands of the objective, the examples, and constraint by their task ID. Doing so reveals the final optimisation problems

$$\left.\begin{aligned} \min_{w_t,b_t,\xi\geq 0} \|w_t\|^2 + C\sum_j \xi_j & \\ \langle w_t, \Phi(\Lambda_j^+)\rangle + b_t \geq -\xi_j \qquad & \forall \Lambda_j^+ \in \mathbb{L}^+,\ q(\Lambda_j^+) = t \\ \langle w_t, \Phi(\Lambda)\rangle + b_t \leq -\Delta(\Lambda) \quad & \forall \Lambda \in \mathbb{L}\setminus\mathbb{L}^+,\ q(\Lambda) = t \end{aligned}\right\} \quad \forall t \tag{6}$$

that can be solved in isolation.

Discussion. We build on a well accepted ranking formulation [4], while the decomposition exploits three reasonable assumptions that we discuss now. First, step 1 requires the loss to be separable in the two argument, *i.e.*, $\Delta(\Lambda_j^+, \Lambda) = u(\Lambda_j^+) - v(\Lambda)$. Such losses arise in practise (*e.g.* [11]), but this is not always the case (*e.g.*, slack-rescaling). Second, step 1 also requires that the constraint set includes *all* pairs of positive/negative examples. This is the natural approach in an *object-class* detection setup, since we do *not* distinguish particular object instances. Third, step 2 exploits the per-task bias terms. Choosing an appropriate function class (over which we optimise) is legitimate option when modelling a problem. In summary, we make three reasonable assumption that allow to transform a structured (ranking) SVM problem into multiple simpler problems which are similar to binary SVMs.

2 VOC 2007 AP computation artefact

The VOC 2007 implementation [2] for computing the average precision (AP) score has an artefact that manifests itself in the low AP regime.[1] In particular, AP scores around 10% have to be read with care: a method with lower AP can in fact be better (*c.f.* Fig. 1). In such cases, looking at the results of all challenge contestants (not only the per-class best ones), we found that AP$<$10% are not uncommon. Moreover, the (apparently big) gap to the best method is often due to the artefact that we will detail now.

As a matter of fact, an AP of $1/11 \approx 9\%$ is obtained if the best-scoring detection is a true positive even if it is the only one or all other detection are false positives. This effect is particularly visible in the case of class *bird* (see Fig. 1) and the reason for the large variance of AP in as can be seen on the official VOC website [2]. MPI_ESSOL and UoCTTI achieve much higher AP than the other contestants, although their precision recall curves (PRC) appear even worse than those of others (*e.g.*, INRIA_PlusClasses). This phenomenon is explained by the fact that these methods' best detection is a true positive (*i.e.*, their PRC start with precision=1, *e.g.*, blue curve in the figure).

The artefact is due to a coarse sampling used to compute the AP. The matlab code fragment (directly copy-pasted from the VOCdevkit, VOCevaldet.m) is listed in Fig. 2. For the sake of brevity, we omit the code that identifies true/false positive detections (among all reported ones) and only list the final part that computes the AP. The input are two arrays reporting precision (prec) and recall (rec) for various detection-score thresholds. The artefact discussed earlier is due to the first sampling position $t = 0$. In that case *line 5* becomes $\max(prec(rec >= 0))$ which is the highest precision ever reached. Hence, if the best-scoring detection is a true positive this value is 1 and the contribution in *line 9* is the above mentioned

[1] The VOC evaluation score changed in 2010 [3], which removes the artefact described here. We chose to use the 2007 version as it enables the comparison with the published results.

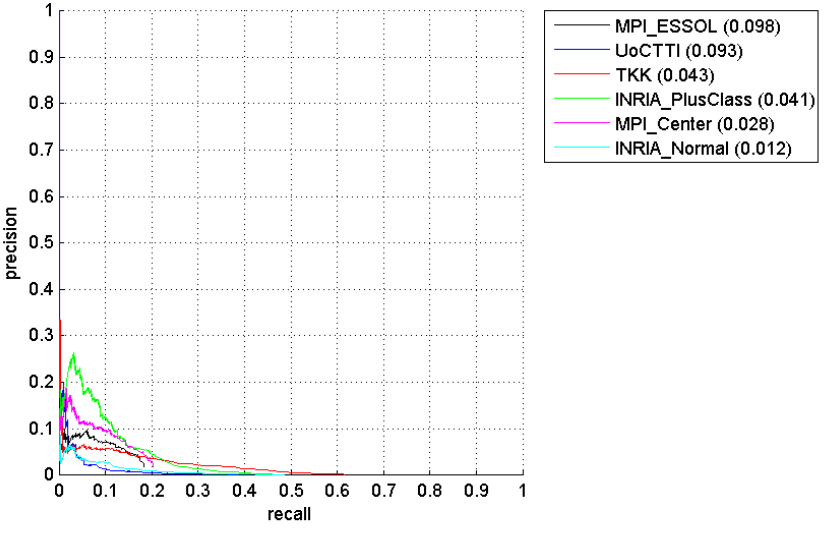

Figure 1: Evaluation artefact. Official challenge results [2] for the class "bird". INRIA_PlusClasses (green) is probably the best performing detector in this plot. However, according to the AP score, MPI_ESSOL (black) seems to be the top method. The plot shows that MPI_ESSOL greatly benefits from the evaluation artefact.

```
1    % compute average precision
2    ap=0;
3    for t=0:0.1:1
4       p=max(prec(rec>=t));
5       if isempty(p)
6          p=0;
7       end
8       ap=ap+p/11;
9    end
```

Figure 2: Matlab code fragment from VOCdevkit 2007

$1/11 = 9\%$. Thus, getting the best-scoring detection correct makes a large difference in situation where the overall precision is quite low (as *e.g.* in the case of "bird"). For system with larger AP this artefact seems not to have a crucial influence anymore. However, it is not unlikely that this coarse sampling still also has some effects in such cases. We did not analyse that in more detail though.

In conclusion, AP scores around 10% have to be read with care due to an implementation artefact.

References

[1] M. B. Blaschko and C. H. Lampert. Learning to localize objects with structured output regression. In *ECCV*, 2008.

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

[3] 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.

[4] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *ICML*, 2004.