Learning a Category Independent Object Detection Cascade

Esa Rahtu, Juho Kannala, Matthew Blaschko

To cite this version:

Esa Rahtu, Juho Kannala, Matthew Blaschko. Learning a Category Independent Object Detection Cascade. International Conference on Computer Vision, Nov 2011, Barcelona, Spain. pp.1052-1059, 10.1109/ICCV.2011.6126351 . hal-00855735

HAL Id: hal-00855735
https://hal.inria.fr/hal-00855735

Submitted on 31 Jan 2014

Learning a Category Independent Object Detection Cascade

Esa Rahtu, Juho Kannala
Machine Vision Group
University of Oulu, Finland

Matthew Blaschko
Visual Geometry Group
University of Oxford, UK

Abstract

Cascades are a popular framework to speed up object detection systems. Here we focus on the first layers of a category independent object detection cascade in which we sample a large number of windows from an objectness prior, and then discriminatively learn to filter these candidate windows by an order of magnitude. We make a number of contributions to cascade design that substantially improve over the state of the art: (i) our novel objectness prior gives much higher recall than competing methods, (ii) we propose objectness features that give high performance with very low computational cost, and (iii) we make use of a structured output ranking approach to learn highly effective, but inexpensive linear feature combinations by directly optimizing cascade performance. Thorough evaluation on the PASCAL VOC data set shows consistent improvement over the current state of the art, and over alternative discriminative learning strategies.

1. Introduction

In this work, we propose a methodology for designing efficient and accurate cascade layers. This enables the replacement of sliding window sampling strategies with an object-aware technique for proposing candidate windows. At the heart of most object detection methods is a discriminant function that distinguishes between windows containing an object of interest and those that contain no object. For practical application of these systems in real-time settings, or to internet scale data, this discriminant function can be the main computational bottleneck in a system. Better discrimination often comes at the expense of computation, be it a result of additional computed features or more expensive function classes [23]. In order to counter this problem, cascade architectures have long been popular in object detection [24, 20, 4, 23, 12]. These work by recognizing that the vast majority of windows in an image will not specify an object bounding box. Consequently, inexpensive classifiers with relatively high false positive rates may nevertheless filter a large majority of bounding boxes while maintaining a very low false negative rate. In this way, a layer of a cascade may filter candidates by an order of magnitude at low cost. Though more computation is needed for true positives, the expected computation per image may be reduced drastically. We follow the general setup of [1] and design cascade layers that learn objectness independent of specific categories.

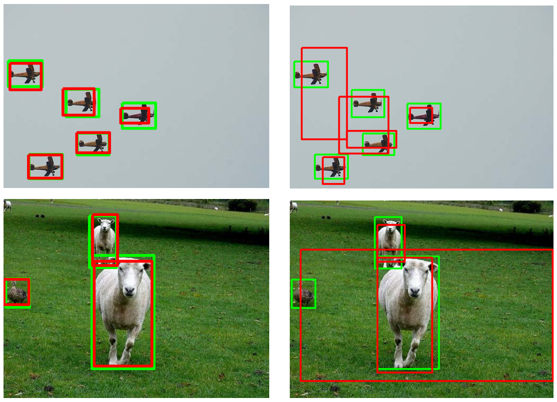


Figure 1. Example detections when returning 100 boxes with the proposed method (left) and the method by Alexe et al. [1] (right). The best detection for each ground-truth box (green) is shown.

We make multiple contributions to cascade design, yielding substantial improvements to the state of the art in generic object cascades. We develop (i) an informative and robust objectness prior from which we sample initial candidate windows, (ii) improved objectness features *at reduced computational cost* to those proposed in [1] to learn a cascade layer, and (iii) a structured output ranking objective to learn a linear discriminant that directly optimizes cascade performance. The initial candidate window selection and resulting discriminant function substantially outperform the state of the art on the VOC 2007 data set [10]. Example detections are shown in Figure 1.

1.1. Related Work

Cascades have been used frequently in the object detection literature. Perhaps most famously, Viola and Jones trained a classifier using boosting, and post hoc ordered the selected weak classifiers into a cascade [24]. Recent work has extended this approach to the multi-view setting [19]. A similar approach was proposed for ordering the evaluation of support vectors [20]. A line of research by Rehg and coauthors considered cascade design in the context of feature selection and asymmetric costs [25, 4].

Torralba et al. proposed to improve object detection with a boosting approach by sharing features across classes [21]. Similarly, Opelt et al. made use of a shared shape alphabet to reduce the complexity of object detection [18]. Felzenszwalb et al. proposed an extension to their pictorial structures model that post hoc proposed detection thresholds to build an efficient parts cascade [12]. Vedaldi et al. took a different approach by training multiple classifiers with different test-time computational costs and arranging them into a cascade [23]. This and the work of Rehg and coauthors marks an important departure from previous cascade work in that the classifiers were trained specifically for performance in a classification cascade rather than being the result of post hoc cascade construction. Ferrari and coauthors have proposed the use of generic objectness measures [1], and have extended this work for simultaneous detection and learning of appearance [8]. Endres and Hoiem have extended this approach to superpixel proposals [9]. The discriminative training of [26] is perhaps the most closely related approach to our method, and uses a very similar objective for cascade optimization. Also, a recent method for creating superpixel object proposals was introduced by Carreira et al. in [5].

Our objectness features are based on superpixel segmentation [11], and share similarities to the superpixel combination techniques of [17]. Our inexpensive, but highly effective objectness features ensure that the proposed method substantially improves over sliding window sampling strategies, both in accuracy and computational cost.

In contrast to much of the previous cascade literature, our work does not design a cascade post hoc, nor do we make parametric assumptions about the errors of a classifier. Rather, we use a non-parametric structured output approach to directly minimize the regularized empirical risk of a single cascade layer. We further apply this in the generic objectness setting, resulting in object location proposals that can subsequently be used for a large number of generic object detection systems. This enables systems to scale to large numbers of object classes, with subsequent layers of the cascade using sophisticated, computationally expensive discriminant functions.

2. Overview of the algorithm

The proposed method consists of three main stages: (i) construction of the initial bounding boxes, (ii) feature extraction, and (iii) window selection.

In the first stage, we generate an initial set of about 100000 tentative bounding boxes based on image specific superpixel segmentation and a general category independent bounding box prior learnt from training data. We show that, by choosing the initial boxes in a correct way, we are able to restrict all further analysis to about 10^5 image windows while loosing only a few correct detections.

In the second stage, we extract objectness features from initial windows. We use three new features, proposed in this paper, as well as the superpixel straddling (SS) feature from [1]. SS cue is used because it can be computed with a relatively small overhead as we compute superpixel segmentation anyway for the new features and initialization. All features together form a four-dimensional vector describing the objectness of the given image subwindow.

In the last stage, we select the final set of bounding boxes (e.g. 100 or 1000) based on an objectness score, which is evaluated as a linear combination of the four feature values. The feature weights for the linear combination are learnt by using a structured output ranking objective function.

In the following three sections we describe the details of the three main stages of our approach. In Section 6, we compare results with the current state-of-the-art method [1].

3. Creating initial bounding boxes

Generating a set of initial bounding boxes is the first stage in our approach. Reducing the set of possible boxes at an early stage is motivated by the fact that it is not feasible to score all subwindows of an image. Although there are efficient subwindow search methods that can avoid explicit scoring of windows in some cases [16], they are limited to certain features and classifiers, and often it may be better to preselect a large enough set of tentative windows [23, 1], as in conventional sliding window methods [24]. However, this preselection greatly affects the final detection result, and it is not always a simple task, especially in the case of generic objects with widely varying aspect ratios.

In order to reduce the number of evaluated windows, many approaches use either a regular grid or sampling [23]. Sampling can be uniform or image-specific [7, 1]. Alexe et al. [1] build a dense regular grid in the four-dimensional window space, evaluate a saliency score for all windows in the grid [13], and finally sample 100000 windows according to the saliency scores. This approach requires evaluating the saliency of millions of windows.

We propose a method that avoids scoring millions of windows. Instead, we compose the initial set of bounding boxes from two subsets: (i) superpixel windows (including the bounding boxes of single superpixels plus those of connected pairs and triplets), and (ii) 100000 windows sampled from a prior distribution which is learnt from annotated multi-class object boxes. The details are as follows. We use superpixels [11] to generate a subset of initial windows because superpixel segmentation usually preserves object boundaries. In fact, as superpixels divide an image into

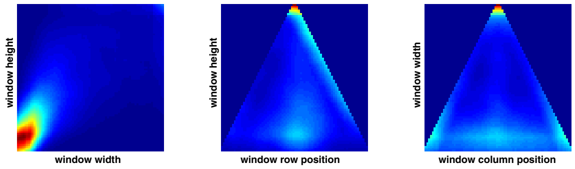


Figure 2. Learnt distributions of object boxes: height versus width, height versus row location, and width versus column location.

small regions of uniform color or texture, as in Fig. 3 (middle), objects are often oversegmented into several superpixels. Hence, it might be tempting to take the bounding boxes of all superpixel combinations as initial windows. However, as we do not want too many windows, we only take the bounding boxes of individual superpixels plus the boxes of connected (i.e. neighboring) superpixel pairs and triplets. Typically this results in a few hundred windows per image.

The vast majority of our initial windows are created by sampling 10^5 boxes from a generic bounding box prior that is learnt by using 15662 objects from PASCAL VOC dataset [10]. Since a subwindow is defined by four coordinates that determine its top-left and bottom-right corners, estimating a 4D density function would be the most straightforward way of learning the prior. However, as the samples are scarce for an accurate estimation of a 4D distribution, we make assumptions about the conditional independence of objects size and location and model their joint density in the form

$$p(a, b, c, r) = p(c|a)p(r|b)p(a, b), \quad (1)$$

where $a, b, c, r \in [0, 1]$ refer to the normalized bounding box width and height, and the column and row coordinates of its centroid, respectively. The normalized column and row coordinates are obtained by dividing the original coordinates by image width and height, respectively.

Thus, it is sufficient to estimate just 1D and 2D distributions for which we have enough data. In practice, $p(a, b)$, $p(r|b)$, and $p(c|a)$ are estimated by collecting three 80×80 histograms: object width versus height, object height versus row location, and object width versus column location. The estimated histograms are smoothed with a Gaussian kernel to enhance their generalizability and the results are shown in Fig. 2 (note the cut-off effect due to image borders).

Given the 2D histograms of Fig. 2, it is straightforward to sample windows from (1). The width and height are sampled from $p(a, b)$, and then, given a and b, the row and column locations are sampled from the corresponding 1D distributions $p(r|b)$ and $p(c|a)$.

4. Features

In this section, we propose three new image features which can be used to characterize the likelihood that a particular rectangular image region is a bounding box of an object. The first feature is based on superpixels [11] and the other two features utilize image edges and gradients.

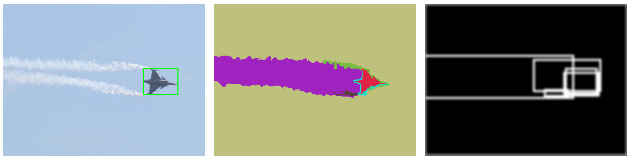


Figure 3. Left: An image and an annotated bounding box. Middle: Superpixel segmentation. Right: A smoothed version of a binary image that shows the bounding boxes of superpixels.

4.1. Superpixel boundary integral (BI)

Superpixels have been shown to be strong cues about object boundaries [11, 1]. For example, Alexe et al. [1] proposed a superpixel-based objectness measure, called superpixel straddling (SS), and used it for detecting generic objects from images. The SS measure has values in the interval [0,1] and it is highest for windows whose boundaries tightly align with the superpixel boundaries. According to the experiments in [1], superpixel straddling is a powerful cue to characterize the likelihood that a certain image window is a bounding box of an object.

We propose another superpixel-based objectness measure, called superpixel boundary integral (BI), which also performs well and is faster to evaluate than superpixel straddling. Our measure is computed from the superpixel bounding boxes instead of the original superpixels. That is, given the bounding boxes of the original superpixels, we construct a binary image that represents the boundaries of the bounding boxes, smooth it, and then define our measure $BI(y)$ for a particular window y as the integral of intensities of the smoothed image along the window boundary. In detail,

$$BI(y) = \frac{\sum_{p \in \mathcal{B}(y)} I_{\mathrm{S}}(p)}{perimeter(I_{\mathrm{S}})}, \quad (2)$$

where I_{S} is a Gaussian smoothed version of the binary image representing superpixel bounding boxes, $\mathcal{B}(y)$ is the set of boundary pixels of y, and the denominator is the perimeter of the entire image in pixels. Thus, $BI(y) \in [0, 1]$, as the upper bound for intensity values in I_{S} is 1 by definition. An example of I_{S} is illustrated in Figure 3 (right).

The proposed BI measure, defined by (2), is efficient to evaluate. Given I_{S} and a window y, $BI(y)$ is simply the sum of intensities of I_{S} over the boundary pixels of y divided by the image perimeter. Moreover, by precomputing the cumulative sums of the rows and columns of I_{S}, the sum in the numerator can be computed with just four subtractions and three additions per window, i.e., one subtraction per bounding line segment. Thus, while BI measure needs only eight operations per window, the number of operations per window required by SS is about seven times the total

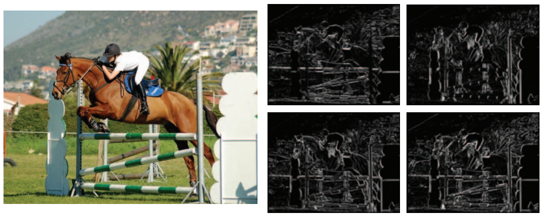


Figure 4. Left: Original image. Right: Edge-weighted gradient magnitude maps for four main orientations.

number of superpixels. In addition, SS requires precomputation of an integral image for each superpixel [1].

4.2. Boundary edge distribution (BE)

The second feature that we propose is based on image edges and gradients and it measures the distribution of oriented edges near the boundary of a window. Given a set of windows $\mathcal{Y}$, our new boundary edge measure (BE) provides a score $BE(y, \mathcal{Y}) \in [0, 1]$ for each window $y \in \mathcal{Y}$. Thus, instead of scoring windows independently, we score windows in a set so that the scoring provides an ordering of windows relative to the set.

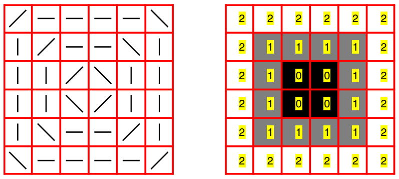
The details for computing the BE measure are as follows. First, for each window y, we partition the window area into non-overlapping rectangular subregions and, in each subregion y_l, we integrate the magnitudes of color gradients of a particular orientation along the image edges. Then, we compute a weighted sum of the integrals over all subregions and divide this sum with its maximum value over all the windows. Thus, $\max_{y \in \mathcal{Y}} BE(y, \mathcal{Y}) = 1$.

Mathematically,

$$BE(y, \mathcal{Y}) = \frac{1}{M(\mathcal{Y})} \sum_{l=1}^{L} \gamma_l \sum_{p \in y_l} G_{d_l}(p), \tag{3}$$

where $M(\mathcal{Y})$ is the maximum of the above double sum over all windows in $\mathcal{Y}$, γ_l is the weight for subwindow y_l, L is the total number of subwindows in the partition of y and $G_{d_l}(p)$ is the edge-weighted gradient magnitude in direction d_l at pixel p. In our case, we have quantized gradient orientations into four bins, i.e. $d_l \in \{1, 2, 3, 4\}$, which correspond to horizontal (0°), vertical (90°), and diagonal (±45°) directions.

The edge-weighted gradient magnitude maps $G_1, \ldots, G_4$ are illustrated in Figure 4. In order to compute each G_k for a given image, we first run a Canny edge detector for the original image and compute its intensity gradient. Thereafter, only gradients of edge pixels contribute to G_k. That is, the gradient magnitude at an edge pixel is divided into the orientation bins of maps G_k proportionally to the cosine of the angle between the gradient direction and bin's reference direction. Finally, the gradient magnitude maps G_k are smoothed by Gaussian filtering to get the results shown in Figure 4.



/	—	—	—	—	\
│	/	—	—	\	│
│	│	/	\	│	│
│	│	\	/	│	│
│	\	—	—	/	│
\	—	—	—	—	/

2	2	2	2	2	2
2	1	1	1	1	2
2	1	0	0	1	2
2	1	0	0	1	2
2	1	1	1	1	2
2	2	2	2	2	2

Figure 5. Window partition into 36 subregions. Left: Normal vector orientations for gradients considered in each subregion. Right: The weights for gradient magnitudes (γ_l).

In our implementation, we divide the image windows into 36 subwindows in a regular 6 × 6 grid. Thus, in our case $L = 36$, and the weights γ_l and the orientations d_l considered in different subwindows are illustrated in Figure 5. As the figure shows, our BE measure aims to capture the closed-boundary characteristics of object windows by assigning the largest weights for gradients that are close to the window boundary and orthogonal to it.

If the number of windows in $\mathcal{Y}$ is large and the windows are partially overlapping, the BE measure can be computed efficiently by precomputing the integral images of maps $G_1, \ldots, G_4$. Then, the inner sum in (3) can be computed by using just four additions or subtractions per window. Thus, the total number of elementary operations per window is about $6L$, i.e. 216. Although our BE feature requires more computation than the BI measure introduced in Section 4.1, it is still very efficient. For example, the CC cue in [1] computes the Chi-square distance between two high-dimensional histograms for every window (dimension 2048), and also the number of integral images that must be precomputed is much higher than in our case.

4.3. Window symmetry (WS)

In addition to the closed boundary property, internal symmetry is another common property of object windows. We utilize it by introducing a window symmetry feature (WS), which measures symmetry across the horizontal and vertical central axes of image windows. Our WS feature is based on the same edge-weighted orientation-specific gradient magnitude maps ($G_1, \ldots, G_4$) as the BE feature. The computational details are described in the following.

Given a set of image windows $\mathcal{Y}$, the symmetry feature $WS(y, \mathcal{Y})$ is evaluated for all $y \in \mathcal{Y}$ as follows. We divide each window y into 16 subwindows in a regular 4×4 grid. Then, in each subwindow, we compute a four-dimensional gradient orientation histogram by integrating the magnitudes from maps G_k within the subwindow, i.e., each G_k corresponds to one histogram bin. Further, as the 4×4 grid divides the main quadrants of the window into 2×2 blocks, we concatenate the four histograms in each quadrant into a one histogram of length 16. Thus, in total, we get four histograms, one per each quadrant of the original window.

Then, we compare pairs of histograms from horizontal (or vertical) neighbor quadrants via histogram intersection in which either one of the histograms is transformed by a mirror reflection across the horizontal (or vertical) central axis. In such a transform the histogram bins corresponding to diagonal and anti-diagonal orientations are swapped and also the histogram blocks originating from a 2×2 grid are swapped according to the mirror reflection axis. In total, we get histogram intersection for four pairs and, finally, we sum these four values together and divide the sum with its maximum value over all the windows in the set $\mathcal{Y}$.

In summary,

$$WS(y,\mathcal{Y}) = \frac{(\iota_{\mathrm{A}}|\bar{\iota}_{\mathrm{B}}) + (\iota_{\mathrm{A}}|\tilde{\iota}_{\mathrm{C}}) + (\iota_{\mathrm{D}}|\bar{\iota}_{\mathrm{C}}) + (\iota_{\mathrm{D}}|\tilde{\iota}_{\mathrm{B}})}{N(\mathcal{Y})}, \quad (4)$$

where the histograms ι_{A}, ι_{B}, ι_{C}, and ι_{D} correspond to the top-left, top-right, bottom-left and bottom-right quadrants of window y, respectively, $(\cdot|\cdot)$ denotes histogram intersection, and the denominator $N(\mathcal{Y})$ is the maximum value of the numerator over all $y \in \mathcal{Y}$. The bar $\bar{\cdot}$ and tilde $\tilde{\cdot}$ denote histogram reflection across window's vertical and horizontal central axis, respectively.

The WS measure defined above can be efficiently evaluated by using integral images of maps G_k. In this case, computing the four-dimensional histograms for the16 sub-windows requires $4^4 = 256$ operations and each intersection of sixteen-dimensional histograms in (4) requires 31 operations, so that the total cost of evaluating (4) is about $(256+4\times 31+5) = 385$ elementary operations per window. Hence, WS measure is not much more complex than BE.

5. Learning feature combinations

5.1. Structured Output Ranking

We propose a learning algorithm that directly optimizes the performance of interest for a cascade: the quality of windows that advance to the next layer of the cascade. We achieve this by modifying the max-margin structured learning framework [22] to enforce ranking constraints that ensure that the windows with the least overlap loss to the ground truth will have higher score than all others:

$$\min_{w,\xi} \frac{1}{2}\|w\|^2 + \sum_i \xi_i \quad (5)$$

$$\text{s.t. } \langle w, \phi_{ij}\rangle - \langle w, \phi_{ik}\rangle \geq \Delta_{ik} - \Delta_{ij} - \xi^i_{jk} \quad (6)$$

$$\forall i, \{j | y^{\omega}_{ij} = 1\}, \{k | y^{\omega}_{ik} = 0\}$$

$$\xi^i_{jk} \geq 0 \qquad \forall i,j,k \qquad \xi_i = \sum_{j,k} \xi^i_{jk} \quad (7)$$

where w is the vector of weights, ϕ_{ij} is a feature vector corresponding to the jth window of the ith image, Δ_{ij} is a corresponding loss, and y^{ω}_{ij} is an indicator variable that selects samples we would like to proceed to the next stage, i.e. samples that should be ranked higher than all others. In this work, we set the indicator variable $y^{\omega}_{ij} = 1 \iff \Delta_{ij}$ is in the best K windows in terms of lowest loss, 0 otherwise. This enforces a margin constraint such that each of the top K windows should be ranked higher than the rest, with a margin proportional to the difference in losses between the two windows. This generalizes standard ranking algorithms to the structured output case where both the higher ranked and lower ranked windows may have non-zero loss.[1]

We base our loss function on the VOC overlap score, $\frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})}$, where B_p is a predicted box and B_{gt} is a ground truth box. While all monotonically decreasing functions of the VOC overlap score are possible loss functions, we have chosen perhaps the most simple one: one minus the overlap score [2]. The exact choice of loss function should be based on application specific overlap tolerances.

This form of learning objective has significant advantages for learning as compared to methods that directly predict the fitness of an individual output y_{ij}, especially for inexpensive but weak features. This is because the loss allows for mistakes to be made for the ordering of the best K windows, *so long as those are the windows that advance to the next layer of the cascade*. It may be easier to discriminate the best windows from those with higher loss than it is to predict the actual fitness of every window. As subsequent layers of the cascade will have access to more sophisticated features and function classes, we may defer to later layers to make this more difficult distinction. We show in the results section that this gives a strong empirical improvement over learning a ranking function that enforces only the ground truth to be ranked higher than other samples.

We use a cutting plane approach to optimizing Equation (5). This requires computing the most violated constraints, which we achieve using a 1-slack optimization approach per image [15, 3]. Algorithm 1 is related [3] and computes this argmax for our objective. This algorithm is linear in the number of candidate windows using bucket sort, resulting in very fast optimization times.

5.2. Ridge Regression

In order to test the hypothesis that the Structured Output Ranking Cascade objective performs better than one that directly predicts the fitness of a given window, we compare its performance to that of large-scale ridge regression. Our implementation is equivalent to training ridge regression on all windows in all images in the training set

$$w = (X^T X + \lambda I)^{-1} X^T (e - \Delta), \quad (8)$$

where e is a vector of ones and Δ is a vector of window losses, and works by computing intermediate matrix vector products one image at a time (i.e. matrix X that contains the features of all windows in all images is not explicitly constructed). The amount of memory used is therefore bounded

[1] We recover the classical ranking SVM [14, 6] in the case that all losses are in $\{0, 1\}$ and there is exactly one training sample.

Algorithm 1 Finding maximally violated constraint for structured output ranking cascades.

```
Ensure: Maximally violated constraint for image $i$ is $\delta_{iy} - \langle w, \Psi_{iy} \rangle \leq \xi_i$
  for all $\{j | y^{\omega}_{ij} = 1\}$ do
    $s^+_j = \langle w, \phi_{ij} \rangle + \Delta_{ij}$
  end for
  for all $\{k | y^{\omega}_{ik} = 0\}$ do
    $s^-_k = \langle w, \phi_{ik} \rangle + \Delta_{ik}$
  end for
  $(s^+, p^+) = \text{sort}(s^+)$, $(s^-, p^-) = \text{sort}(s^-)$
  $j = 1, \delta_{iy} = \Delta_{i+} = 0, \Psi_{iy} = \phi_+ = \mathbf{0}$
  for all $k$ do
    while $s^-_k > s^+_j \wedge j \leq n_+ + 1$ do
      $\phi_+ = \phi_+ + \phi_{ip^+_j}$
      $\Delta_{i+} = \Delta_{i+} + \Delta_{ip^+_j}$
      $j = j + 1$
    end while
    $\Psi_{iy} = \Psi_{iy} + \phi_+ - (j-1)\phi_{ip^-_k}$
    $\delta_{iy} = \delta_{iy} + (j-1)\Delta_{ip^-_k} - \Delta_{i+}$
  end for
```

and it is feasible to apply to the hundreds of millions of windows resulting from a typical training set of several thousand, or even millions of images.

5.3. Non-maxima suppression

Given a large scored set of tentative windows for an image, the final task is to select a smaller representative subset of windows which would contain the bounding boxes of all the objects in the image. In order to succeed in this task, it is not sufficient to simply select the best scoring windows but some kind of a non-maxima suppression is needed. In fact, choosing simply the best scoring windows could lead to a situation where certain salient image regions are covered with multiple redundant windows and other regions are totally uncovered, which implies poor recall rates.

Our approach to non-maxima suppression has two stages. First, we notably reduce the set of candidate windows by selecting a certain number (e.g. 10000) of such windows that possess a local score maxima. Second, this reduced set of windows is used as a pool from which we select the final number (e.g. 1000) of windows by a similar approach as in [23]. The details are as follows.

In the first stage, we partition the four-dimensional space of image windows into a regular grid of volume elements (voxels) at multiple resolution levels, and search for such windows that generate local score maxima in the voxel grid, starting from the lowest resolution grid and continuing until we have found a given number of maxima (10000 in our case). This can be done efficiently so that the complexity of the process is only linear in the number of initial windows.

Second, given the reduced set of candidate windows, we select the final windows using a similar procedure as in [23]. That is, we sort the scores of the candidate windows in descending order, select the best scoring window and continue to select additional windows in the score order, but ensuring that the overlap of a newly selected window with any of the previously selected ones does not exceed a threshold τ. Although this could be time consuming with a large number of windows, efficiency is not a problem in our case due to the first selection stage above, which acts as a prefilter.

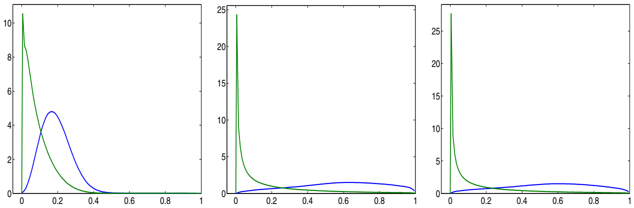


Figure 6. Distribution of feature values for boxes whose overlap score with a ground truth box is ≥ 0.5 (blue) and < 0.5 (green): BI (left), BE (middle), and WS (right).

6. Experiments

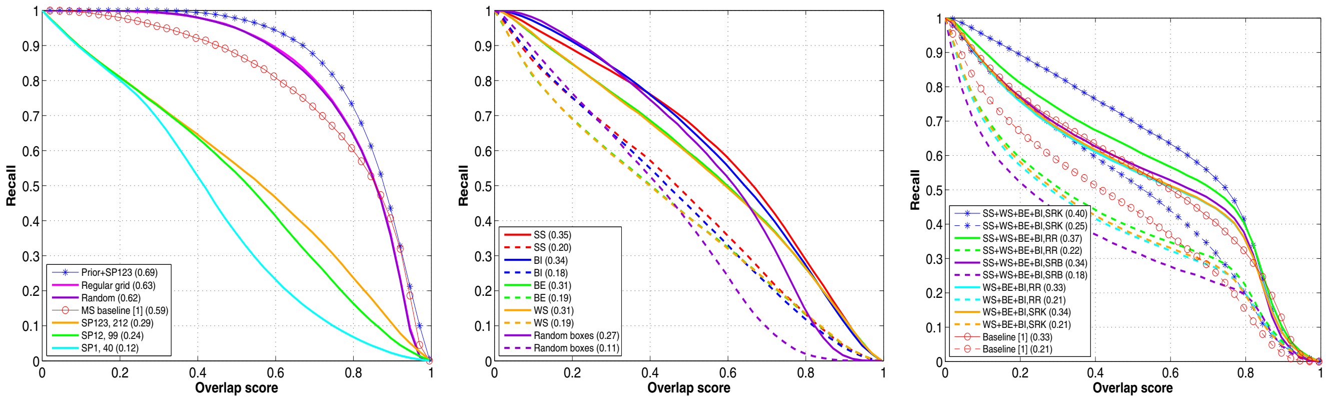
We experiment with PASCAL VOC 2007 dataset [10], which contains 2501, 2510, and 4952 images for training, validation, and testing, respectively. The images are annotated with ground-truth bounding boxes of objects from 20 classes. Some objects are marked with labels *difficult* or *truncated* but they are also included in our evaluation. We use objects from both the training and validation sets to learn the prior of Section 3. The weights for the linear feature combination in the final objectness score are learnt from the training set of [1] (50 images).

The detection performance is measured using a recall-overlap curve, which indicates the recall rate of ground truth boxes in the test set for a given minimum value of the overlap score [23]. We also report the area under the curve (AUC) between overlap scores 0.5 and 1, and normalize its value so that the maximum is 1 for perfect recall. The overlap limit 0.5 is chosen here since less accurately localized boxes have little practical importance.

6.1. Initial window experiments

In the first experiment we evaluate initial windows by computing the recall-overlap curves for sets of 10^5 windows per image. In particular, we compare our windows to the initial window set by Alexe et al. [1], which is referred to *MS baseline* in Fig. 7(a) and computed using their code. We also show the curves obtained with uniform sampling (*Random*) and regular grid of 10^5 boxes (*Regular grid*).

Our set of initial windows is illustrated by the blue curve in Fig. 7(a) (*Prior+SP123*). We additionally illustrate its subsets by three curves: (i) bounding boxes of superpixels (*SP1*), (ii) bounding boxes of superpixel singletons and connected pairs (*SP12*), and (iii) bounding boxes of superpixel singletons, and connected pairs and triplets (*SP123*). Fig. 7(a) also reports the AUC values (in parentheses) and



(a) Initial window sets

(b) Inidividual features

(c) Feature combinations

Figure 7. The resulting *recall-overlap* for (a) initial window, (b) single feature, and (c) feature combination experiments. The number in parentheses, following each method name, denotes the AUC value. In (b) and (c) solid lines refer to 1000 returned boxes and dashed line to 100 returned boxes. RR, SRB, and SRK refer to ridge regression, structured ranking with ground truth, and structured ranking K-best, respectively (see text for details). Our initial sampling and final system performance (blue curves) show substantial improvement over the baseline of Alexe et al. (red curves).

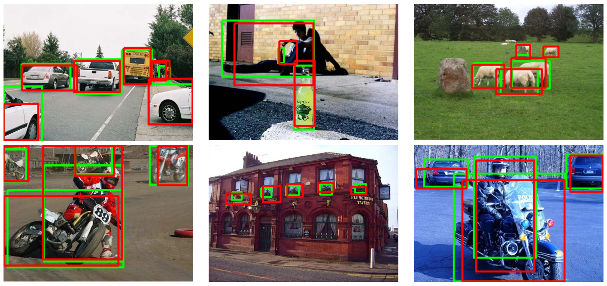


Figure 8. The green boxes show the ground truth and the red ones show the best detections within the returned 1000 boxes.

the average number of boxes per image in the subsets that are based on superpixels.

6.2. Individual feature experiments

In the second experiment, we assess the new features by computing the distributions of feature values for windows whose maximum overlap score with ground-truth boxes is either ≥ 0.5 or < 0.5. The results are in Fig. 6.

We also compare our features to the SS cue by evaluating them for all 10^5 initial boxes and then sampling either 100 or 1000 boxes per image with probabilities that are proportional to the feature values. The corresponding recall-overlap curves are shown in Fig. 7(b).

6.3. Feature combination experiments

The final experiment evaluates the performance of the entire method. We consider two sets of features, $\{WS, BE, BI\}$ and $\{SS, WS, BE, BI\}$, as well as three methods for learning the feature weights: ridge regression (denoted RR in the figure), structured output ranking with ground truth (SRB),[2] and structured output ranking with top K (denoted SRK). The parameter K for structured output ranking was set to 1000.

A baseline for this experiment is set by [1]. The baseline curves are obtained by using the boxes precomputed by the authors of [1] and available online. For all methods we draw two curves corresponding to 100 or 1000 output boxes. The results are shown in Figure 7(c). Figure 8 also shows some example detections using our approach with four features.[3] Finally, it should be noted that the recall rates in Fig. 7 would consistently increase if the truncated objects were ignored, but this would not change the ranking of methods.

7. Discussion

The first experiment compared the different approaches in creating the initial window set. The results in Fig. 7(a) clearly illustrate that the best recalls are achieved using the proposed combination of the learned prior (1) and bounding boxes of superpixels. The baseline methods were outperformed at all overlap scores by up to 15 percent in recall. The improvement is significant considering that this will be the upper bound to the performance of the following cascade levels and that the proposed method requires far less computations than [1].

From Figure 7(b), one notices that the difference between the methods is almost negligible at high overlap levels. However when the overlap drops, the superpixel based cues, SS and BI, seem to perform better than BE and WS. One reason could be that the object boundaries are poorly covered when there is low object overlap.

[2] The objective is the one given in Equation (5), but y_{ij}^{ω} is set to 1 only for ground truth windows.

[3] More examples and precomputed object boxes for PASCAL VOC 2007 dataset are available online at http://www.cse.oulu.fi/MVG/Downloads/ObjectDetection

The feature combination results further illustrate a clear gain compared to the baseline method [1]. The observed difference is up to 12 percent and is most pronounced at overlap levels < 0.8. Performance generally increased with the addition of new features, indicating that they may contain complimentary information for discrimination. Although not shown in Fig. 7(c) due to lack of space, we also computed results for pairwise combinations of the new features (i.e. $BE+BI$, $WS+BI$, $WS+BE$). We found that $BE+BI$ and $WS+BI$ are almost as good as $WS+BE+BI$, and $WS+BE$ is only slightly worse. Thus, we get comparable results to [1] with various pairs of the new features and without using any of the features of [1].

When comparing the learning techniques (ridge regression and structured output ranking), it can be seen that structured ranking performs better than ridge regression. Further, in general we found the ridge regression to be unstable, especially with multiple cues. In contrast, the structured output ranking showed stable behavior whenever new features or training data were added. The K best variant of structured output ranking performs substantially better than the version that requires ground truth to be ranked higher than sampled windows. This confirms our hypothesis that K best ranking is more suited to cascade design as it directly optimizes performance at a given reduction in the number of windows, while leaving the exact ordering of these windows to later cascade layers that will have access to more expensive features and function classes.

8. Conclusions

In this paper, we presented an algorithm for locating object bounding boxes independent of the object category. We follow the general setup of [1] and introduce several substantial improvements to the state-of-the-art generic object detection cascades. The main contributions included new simple approaches in generating the initial candidate windows and constructing the objectness descriptors. Furthermore we build an effective linear feature combinations using a structured output ranking objective. In the experiments we observed over 10 percent improvement in recall rate compared to state-of-the-art approach [1]. Even at overlap accuracy 0.75 more than half of all the annotated objects in VOC 2007 dataset (including difficult and truncated) were captured within a set of 1000 returned candidate windows per image.

Acknowledgements

MBB is funded by a Newton International Fellowship and ER by the Academy of Finland (Grant no. 128975).

References

[1] B. Alexe, T. Deselaers, and V. Ferrari. What is an object? In *CVPR*, 2010. 1, 2, 3, 4, 6, 7, 8

[2] M. B. Blaschko and C. H. Lampert. Learning to localize objects with structured output regression. In *ECCV*, 2008. 5

[3] M. B. Blaschko, A. Vedaldi, and A. Zisserman. Simultaneous object detection and ranking with weak supervision. In *NIPS*, 2010. 5

[4] S. Brubaker, M. Mullin, and J. Rehg. Towards optimal training of cascaded detectors. In *ECCV*. 2006. 1, 2

[5] J. Carreira and C. Sminchisescu. Constrained parametric mincuts for automatic object segmentation. *CVPR*, 2010. 2

[6] O. Chapelle and S. S. Keerthi. Efficient algorithms for ranking with svms. *Inf. Retr.*, 13:201–215, June 2010. 5

[7] O. Chum and A. Zisserman. An exemplar model for learning object classes. In *CVPR*, 2007. 2

[8] T. Deselaers, B. Alexe, and V. Ferrari. Localizing objects while learning their appearance. In *ECCV*, 2010. 2

[9] I. Endres and D. Hoiem. Category independent object proposals. In *ECCV*, 2010. 2

[10] M. Everingham, L. V. Gool, C. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge 2007. 2007. 1, 3, 6

[11] P. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59(2):167–181, 2004. 2, 3

[12] P. F. Felzenszwalb, R. B. Girshick, and D. A. McAllester. Cascade object detection with deformable part models. In *CVPR*, pages 2241–2248, 2010. 1, 2

[13] X. Hou and L. Zhang. Saliency detection: A spectral residual approach. In *CVPR*, 2007. 2

[14] T. Joachims. Optimizing search engines using clickthrough data. In *ACM SIGKDD*, 2002. 5

[15] T. Joachims, T. Finley, and C.-N. Yu. Cutting-plane training of structural svms. *Machine Learning*, 77:27–59, 2009. 5

[16] C. Lampert, M. Blaschko, and T. Hofmann. Efficient subwindow search: A branch and bound framework for object localization. *IEEE TPAMI*, 31(12):2129 –2142, 2009. 2

[17] T. Malisiewicz and A. A. Efros. Improving spatial support for objects via multiple segmentations. In *BMVC*, 2007. 2

[18] A. Opelt, A. Pinz, and A. Zisserman. Incremental learning of object detectors using a visual shape alphabet. In *CVPR*, 2006. 2

[19] X. Perrotton, M. Sturzel, and M. Roux. Implicit hierarchical boosting for multi-view object detection. In *CVPR*, 2010. 2

[20] S. Romdhani, P. Torr, B. Schölkopf, and A. Blake. Computationally efficient face detection. In *ICCV*, 2001. 1, 2

[21] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing visual features for multiclass and multiview object detection. *IEEE TPAMI*, 29:854–869, 2007. 2

[22] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *ICML*, 2004. 5

[23] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *ICCV*, 2009. 1, 2, 6

[24] P. Viola and M. Jones. Robust real-time object detection. *IJCV*, 57(2):137–154, 2002. 1, 2

[25] J. Wu, J. Rehg, and M. Mullin. Learning a rare event detection cascade by direct feature selection. *NIPS*, 2004. 2

[26] Z. Zhang, J. Warrell, and P. Torr. Proposal generation for object detection using cascaded ranking svms. *CVPR*, 2011. 2