How good are detection proposals, really?

Jan Hosang
http://mpi-inf.mpg.de/~jhosang
Rodrigo Benenson
http://mpi-inf.mpg.de/~benenson
Bernt Schiele
http://mpi-inf.mpg.de/~schiele

MPI Informatics
Saarbrücken, Germany

Abstract

Current top performing Pascal VOC object detectors employ detection proposals to guide the search for objects thereby avoiding exhaustive sliding window search across images. Despite the popularity of detection proposals, it is unclear which trade-offs are made when using them during object detection. We provide an in depth analysis of ten object proposal methods along with four baselines regarding ground truth annotation recall (on Pascal VOC 2007 and ImageNet 2013), repeatability, and impact on DPM detector performance. Our findings show common weaknesses of existing methods, and provide insights to choose the most adequate method for different settings.

1 Introduction

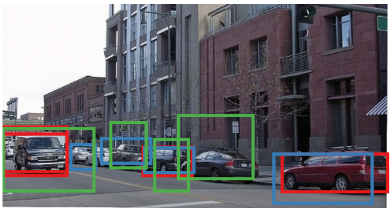


Figure 1: How to evaluate the quality of such detection proposals?

Object detection is traditionally formulated as a classification problem in the well known "sliding window" paradigm where the classifier is evaluated over an exhaustive list of positions, scales, and aspect ratios. Steadily increasing the sophistication of the core classifier has led to increased detector performance [12, 15, 32].

A typical sliding window detector requires $\sim 10^6$ classifier evaluations per image. One approach to overcome the tension between computational tractability and high detection quality, is the notion of "detection proposals" (sometimes called "objectness" or "selective search"). Under the assumption that all objects of interest share common visual properties that distinguish them from the background, one can train a method that, given an input image, outputs a set of detection window proposals that are likely to contain the objects. If high recall can be reached with $\sim 10^4$ or less windows, significant speed-ups can be achieved, enabling the use of even more sophisticated classifiers.

Besides the potential to improve speed, the use of detection proposals changes the data distribution that the classifier handles, thus also potentially improving detection quality (reduce false positives). It is interesting to note that the current two top performing detection methods for Pascal VOC [11] use detection proposals [15, 32].

Contributions In contrast to previous work that typically performs a partial evaluation when introducing a novel method, this paper aims to revisit existing work on detection proposals and compare all publicly available methods in a common framework to better understand their benefits and limitations. We attempt to provide the necessary insights to understand when to choose which method.

Our contributions are the following. (1) We review existing detection proposal methods (§2) and compare them in a unified evaluation. Our evaluation aims to provide an unbiased view of existing methods. (2) We introduce the notion of repeatability for detection proposals, discuss why it matters when considering detection proposals for object detection, and compare repeatability of existing methods (§3). (3) We evaluate overlap with annotated ground truth objects on the Pascal VOC 2007 test set, and for the first time, over the larger and more diverse ImageNet 2013 validation set (§4). This latter experiment aims at detecting possible biases towards the Pascal VOC objects categories. (4) Finally we evaluate the influence on detector performance using different proposal methods (§5). (5) All bounding boxes from our experiments and the evaluation scripts will be released with the paper. The results presented summarise more than 500 experiments over different data sets, totalling to more than 2.5 months of CPU computation.

2 Detection proposal methods

Interestingly, the spirit of detection proposals is similar to the idea of interest point detection [23, 28]. Interest points were proposed at a time when computing feature descriptors densely was computationally too expensive and some selection of interest points was important. Feature descriptors computed around such interest points were used successfully for classification, retrieval, and detection. Today however, with the increase of computing power, it is standard practice to use dense feature extraction instead [27]. Thus we may ask: do object proposals help detection quality, or are they just a transition technology until we have sufficient computing power?

Detection proposal methods are based on low-level image features to generate candidate windows. One can reinterpret this process as a discriminative one; given the low-level features the method quickly decides whether a window should be considered for detection or not. In this sense detection proposal methods are related to cascade methods [8, 18, 31], which use a fast (but inaccurate) classifier to discard the vast majority of unpromising proposals. Although traditionally used for class specific detection, cascade methods also apply to sets of categories [26], and in principle can be applied to a very large set of categories (see `Bing` method below).

Since their introduction [1, 16] multiple works have explored the idea of generating detection proposals. We briefly review all methods we are aware of in chronological order.
`gPbUCM` [16] is a leading method for grouping pixels into objects. Given an input image it hierarchically groups segments and uses these segments (or bounding boxes around them) as object detection candidates [16, 17]. This paper evaluates the bounding box proposals since not all methods generate a segmentation of the proposals.
`Objectness` [1, 2] is a term that refers to a measure of how likely a detection window contains an object (of any category). [2] estimates this score based on a combination of multiple cues such as saliency, colour contrast, edge density, location and size statistics, and how much such windows overlap with superpixel segments.
`CPMC` [4, 5]: Most methods are built upon some form of hierarchical segmentation. `CPMC` avoids this by generating a set of overlapping segments. Each proposal segment is the solution of a binary segmentation problem, initialised with diverse seeds. Up to 10^4 segments are

generated per image, which are subsequently ranked by objectness using a trained regressor.
`Endres2010` [9, 10]: The approach from [10] mixes a large set of cues. It uses a hierarchical segmentation, a learned regressor to estimate boundaries between surfaces with different orientations, graph cuts with different seeds, and parameters to generate diverse segments (similar to `CPMC`). It also learns a ranking function for the proposed segments.
`SelectiveSearch` [29, 30] is a method where no parameters are learned. The authors carefully engineer features and score functions that greedily merge low-level superpixels. The authors obtain state of the art object detections on Pascal VOC and ILSVRC2011.
`Rahtu2011` [24] revisits [2] and obtains a significant improvement by proposing new objectness cues and a different method to learn how to combine them.
`RandomizedPrim's` [22] is similar to `SelectiveSearch` in terms of features to merge low-level superpixels. The key difference is that the weights of the merging function are learned and the merging process is randomised.
`Bing` [6] is one of the only two methods that are not based on segmentation. A simple linear classifier over edge features is trained and applied in a sliding window manner. Using adequate approximations a very fast class agnostic detector is obtained (1 ms/image on CPU).
`MCG` [3] is one of the most recent methods combining `gPbUCM` and `CPMC`. The authors propose an improved multi-scale hierarchical segmentation (similar to `gPbUCM`), a new strategy to generate proposals by merging up to 4 segments, and (similar to `CPMC`) a new ranking procedure to select the final detection proposals.
`Rantalankila2014` [25], also recently proposed, combines `SelectiveSearch` and `CPMC`. Starting from low-level superpixels the authors propose a merging strategy, similar to `SelectiveSearch` but using different features. These merged segments are then used as seeds for a `CPMC`-like process to generate larger segments.
`Rigor` [20] is a variant of `CPMC` obtaining higher quality by using different low-level superpixels and different features for merging segments. They also obtain higher speed by minimizing redundant computations.
`EdgeBoxes` [33] is similar in spirit to `Bing`, a scoring function is evaluated in a sliding window fashion. This method uses object boundaries estimates (obtained via structured decision forests) as feature for the scoring. Interestingly, the authors propose tuning parameters to optimize recall at a desired overlap threshold (see section 4).

Recently, Kang et al. [21] proposed a "data-driven objectness" approach. Although it showed promising results for their indoor application scenario, the method seems of limited applicability to other datasets, and thus we do not consider it in our evaluation.

The majority of the methods is based on some low-level segmentation: five use [13], two use a variant of the `gPbUCM` segmentation method, and `Endres2010` uses its own custom low-level segmentation (also used by `Rigor`). Only three methods work without computing low-level segments (`CPMC`, `Bing`, `EdgeBoxes`). At the time of writing no code is available for `gPbUCM` and `Rigor`[1], thus we do not consider these in our evaluation. In total seven out of the ten methods evaluated in this paper use some low-level segmentation, five out of these eight use [13].

It should also be noted that since [1], all related work evaluates the quality of detection proposal methods based on the overlap with ground truth annotations of the Pascal VOC dataset (2007 or 2010) [11]. Although a relevant metric, its repeated use opens the doors for over-fitting to this specific dataset. In this paper we also consider other evaluations.

Number of windows The different methods listed above provide different numbers of de-

[1]Recently published at CVPR 2014.

tection candidates. Some provide rather few windows ($\sim 10^2$), others provide a large number ($\sim 10^5$). Some methods do provide sorted (scored) windows, others do not. Having more windows increases the chance for high recall, thus for each method in all experiments we do our best effort to generate a similar average number of candidate windows per image. See supplementary material for details on how this is achieved for each method.

2.1 Baselines

Besides the above methods, we also consider a set of baselines that serve as quantitative reference points. All of the above candidate methods and the following baselines are class independent.

Uniform: To generate detection proposals, we uniformly sample the bounding box parameters centre position, square root area, and log aspect ratio. We estimate the range of these parameters on the Pascal VOC 2007 training set after discarding 0.5% of the smallest and largest values, so that our estimated distribution covers 99% of the data.

Gaussian: We also estimate a multivariate Gaussian distribution for the bounding box parameters centre position, square root area, and log aspect ratio. After calculating mean and covariance on the training set we sample proposals from this distribution.

SlidingWindow places windows on a regular grid as common for sliding window object detectors. The contingent of the requested number of proposals is distributed across different windows sizes. For each window size, we place the windows uniformly across the image. The procedure is inspired by the implementation of Bing [6].

Superpixels: As discussed in the next sections, low-level superpixels have an important influence on the behaviour of the detection proposal methods. As five of the compared methods build on [13], we use it as a baseline: each low-level segment is used as a detection proposal. We should expect this method to have low recall for objects (§4), but high repeatability (§3).

3 Proposal repeatability

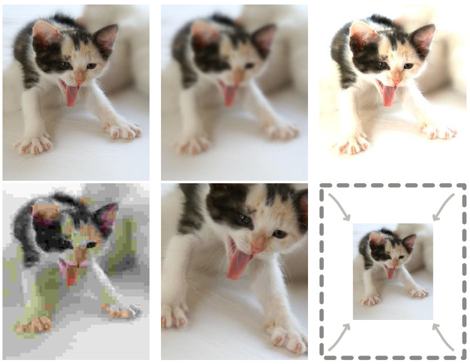


Figure 2: Example of the image perturbations considered. Top to bottom, left to right: original, then blur, illumination, JPEG artefact, rotation, and scale perturbations.

Before looking into how well the different object proposals overlap with ground truth annotations of objects, we want to answer a more basic question. Training a detector on detection proposals rather than on all sliding windows modifies the distribution of negative windows that the classifier is trained on. If the proposal method does not consistently propose windows on similar image content without objects or with partial objects, the classifier cannot produce useful scores on negative windows on the test set. We call the property of proposals being placed on similar image content the repeatability of a proposal method. Intuitively the detection proposals should be repeatable on slightly different images with the same image content. To evaluate repeatability we project proposals from one image into another slightly modified image. Pascal does not contain suitable images. An alternative is the dataset of [23], but it only consists of 54 images and only few of the images contain objects. Instead, we opt to apply synthetic transformations to Pascal images, so we know how to project proposals from one image into another.

3.1 Evaluation protocol

Our evaluation protocol is inspired by [23], which evaluates interest point repeatability. For each image in the Pascal VOC 2007 test set, we generate several perturbed versions. We consider changes in scale, blur, small rotation, illumination, and JPEG compression (see figure 2). The details of the transformations and additional examples of the perturbed images are provided in the supplementary material.

For each pair of reference and perturbed images we compute detection proposals with a given algorithm (requesting 1 000 windows per image). The proposals are projected back from the perturbed into the reference image and then matched to the proposals in the reference image. All proposals whose centre lies outside the image after projection are removed before matching (which can only happen for rotation). Matching is done greedily according to the intersection over union (IoU) criterion. Given the matching, we plot the recall for every IoU threshold and define the area under this "recall versus IoU threshold" curve to be the repeatability. Methods that propose windows at similar locations at high IoU are more repeatable, since the area under the curve is larger.

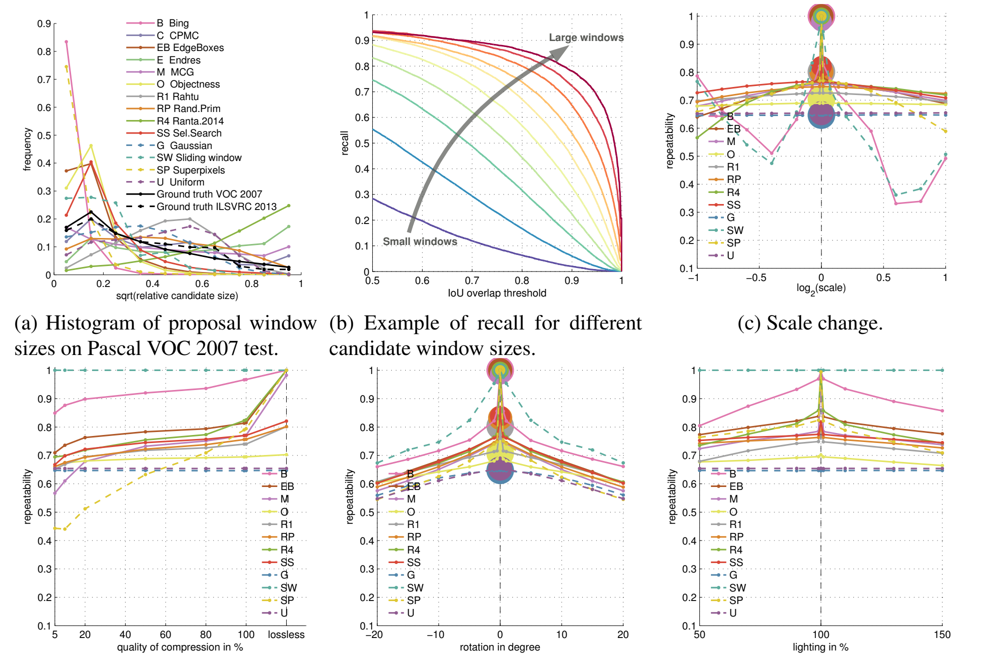
One issue regarding such proposal matching is that large windows are more likely to match than smaller ones. This effect is important to consider since different methods have quite different distributions of window areas (see figure 3a). To reduce the impact of this effect, we bin the original image windows by area (into 10 groups), and evaluate the area under the recall versus IoU curve per size group. The plots in figure 3 show the unweighted average across size bins.

Figure 3a shows that the different proposal methods exhibit a variety of different distributions of proposal window sizes and in general different from the ground truth annotation distribution. This confirms the need for normalisation based on the window size. Figure 3b shows the recall versus intersection over union (IoU) curve for a blur perturbation. This example shows the effect, that large rectangles have higher repeatability, which is why we use the average across size bins to analyse repeatability in the following.

We omit the slowest two methods, `CPMC` and `Endres2010`, from this experiment because it involves running the candidate detector over the Pascal test set ~50 times (once for every perturbation).

3.2 Repeatability results

There are some salient aspects of the result curves in figure 3 that need some explanation. First, not all methods have 100% repeatability when there is no perturbation. This is due to random components in the selection of proposals for several methods (`Bing` being the most notable exception). It is possible to control the random aspect by fixing the random seed, but this would not be effective when perturbations are present and the underlying segmentation changes: a drop in performance could be due to pseudo-random samples or to an actual change in behaviour. Another approach would be to remove the random aspect from all methods, but this would denature the method being evaluated and we would actually evaluate a different algorithm. By keeping the random aspect when no transformation is applied to the image, we can compare values along the change in perturbation.



(a) Histogram of proposal window sizes on Pascal VOC 2007 test.

(b) Example of recall for different candidate window sizes.

(c) Scale change.

(d) JPEG artefacts.

(e) Rotation.

(f) Illumination.

Figure 3: Repeatability results under perturbations. See also supplementary material.

A second important aspect is the large drop of repeatability for most methods, even for very subtle image changes. We have created the `Superpixels` baseline as a means to validate our hypothesis regarding this effect. Most methods use low-level superpixels as a building block for the proposal windows. We suspect that the superpixels themselves are unstable under small perturbations. This is supported by the fact that the `Superpixels` baseline also shows a strong drop (since it is a direct reflection of the superpixels themselves). Since proposals on larger perturbations are all matched against the same proposals in the reference (unperturbed) image, after the very first small perturbation most windows are "lost", the remaining windows that are correctly matched are progressively lost as the perturbation level increases. Inversely we notice that methods that are not based on superpixels are most robust to small image changes (`Bing` and the baselines that ignore image content being the most noticeable ones).

We now briefly discuss the effects under each perturbation, shown in figure 3:
Scale change 3c: All methods except `Bing` show a drastic drop with small scale changes, and minor relative degradation for larger changes. `Bing`'s drop is less extreme for very small scale changes. For larger changes, however, it presents a non monotonous behaviour because it uses a coarse set of box sizes while searching for detection candidates. Such coarse set impedes `Bing` to detect the same object at two slightly different scales. Logically, our `SlidingWindow` baseline suffers from the same effect.
JPEG artefacts 3d: Similar to scale change, even small JPEG artefacts have a large effect and more aggressive JPEG compression factors show monotonic degradation. Despite using gradient information `Bing` is the method most robust to these kind of changes. It should be noted that JPEG images with 100% quality are still lossy. Only when requesting lossless JPEG the superpixels are preserved because the image content does not change.

Rotation 3e: Here all methods are equally affected by the perturbation. The drop of the `Uniform` and `Gaussian` baselines indicate the repeatability loss due to the fact that we are matching rotated bounding boxes.
Illumination change 3f shows a similar trend to JPEG artefacts. We notice here that the gradient information used by `Bing` is more robust to small changes than the superpixels that are used by other methods. Changes in illumination reduce the difference between neighbour pixels, and in turn, affect how they are merged into larger superpixels.
Blurring exhibits a similar trend to JPEG artefacts, although the drop is stronger for a small σ. The corresponding plot can be found in the supplementary material.

Overall it seems that `Bing` and `EdgeBoxes` are more repeatable than other methods, possibly because both use machine components (SVM classifier for scoring and decision forests for features computation, respectively). We also conclude that the sensitivity of superpixels to image perturbations is a major cause for degradation in repeatability of several detection proposal methods.

To better understand how much repeatability influences detector performance, one should train detectors on proposals and compare the detection performance of the detector using different proposal methods. This is, however, a more complicated procedure that involves many new parameters (see e.g. [15, 32]) which is why we leave this for future work.

4 Proposal recall

When using detection proposals it is important to have a good coverage of the true objects in the test image, since missed objects will never be recovered. Thus it is common practice to evaluate the quality of proposals based on ground truth annotations.

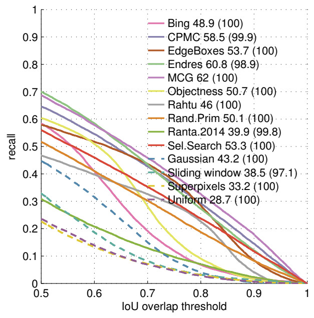
4.1 Evaluation protocol

The protocol introduced in [1] has served as a guideline for most other evaluations in the literature. Typically the detection proposal method is trained on a subset of the Pascal VOC categories, and tested on the test set, including unseen categories. The metric of interest we use is the fraction of ground truth annotation covered above a certain intersection over union (IoU) threshold. Note that evaluating (class agnostic) detection proposals is quite different from traditional class-specific detection [19], since most of the metrics (class confusion, background confusion, precision, etc.) do not apply.

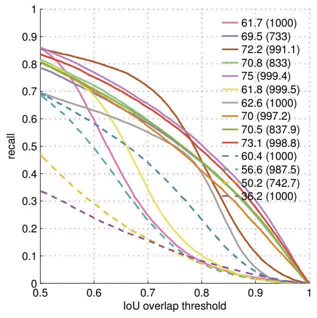
Previous work includes comparisons, however the train and test sets vary amongst papers, and the metrics shown tend to favour different methods. We provide an extensive unified evaluation, that shows how a different perspective can offer different ranking of the methods (see figure 4b versus 5b).

We use the methods as provided by the authors. Different methods may be trained on different sets, but we think this is still fair since we care about absolute quality. Some methods do not have training at all (`SelectiveSearch`), yet provide competitive results. To mitigate this effect we evaluate on the full Pascal VOC 2007 test set, including all categories.

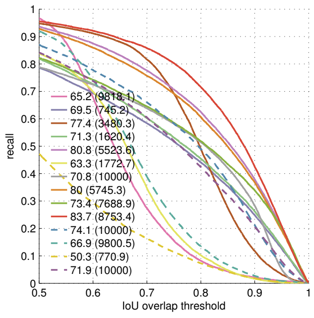
The Pascal VOC 2007 test set has only 20 categories, present in a diverse set of $\sim 5\,000$ unconstrained images, yet detection proposal methods attempt to predict "any" object. On principle, methods which are particularly good at detecting non-Pascal categories will be penalised compared to the ones that are good at these 20 categories but nothing else. To



(a) 100 proposals per image.

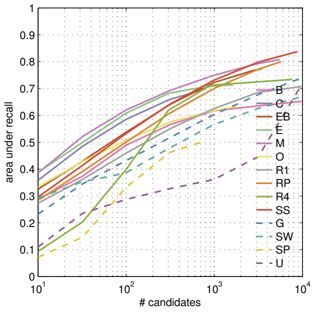


(b) 1 000 proposals per image.

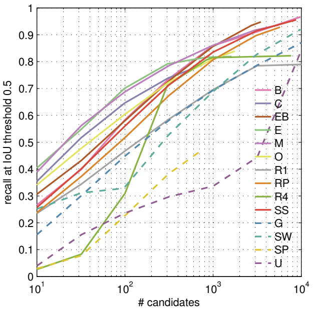


(c) 10 000 proposals per image.

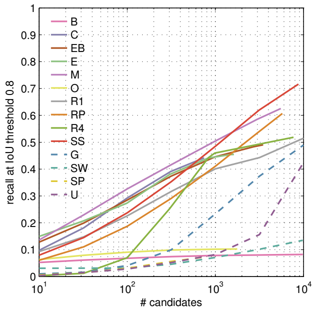
Figure 4: Recall versus IoU threshold on the Pascal VOC 2007 test set. Numbers next to label indicate area under the curve and average number of windows per image, respectively.



(a) Area under "recall versus IoU threshold" curves.



(b) Recall at IoU above 0.5.



(c) Recall at IoU above 0.8.

Figure 5: Recall versus number of proposed windows on the Pascal VOC 2007 test set.

investigate this dataset bias, we also evaluate methods on the larger ImageNet [7] 2013 validation set, which contains annotations for 200 categories over $\sim 20\,000$ images. It should be noted that these 200 categories are *not* fine grained versions of the Pascal ones. It includes additional types of animals (e.g. crustaceans), food items (e.g. hot-dogs), house hold items (e.g. diapers), and other diverse object categories.

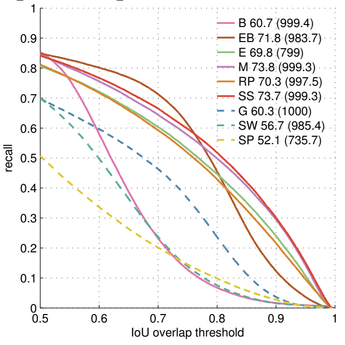


Figure 6: Recall versus IoU threshold, for 1 000 proposals per image on the ImageNet 2013 validation set. See figure 4 and supplementary material.

4.2 Recall results

Results in figure 4 present a consistent trend across the different metrics. `MCG`, `EdgeBoxes`, and `SelectiveSearch` seem to be the best methods across different number of proposals.

We show `EdgeBoxes` when tuned for IoU at 0.7, which results in a clear bump in recall. When tuned for IoU $= 0.9$, for 10^3 proposals, `EdgeBoxes` is below `MCG`, and when tuned for IoU $= 0.5$ it reaches about 93% recall. `MCG` shows the most consistent results when varying the number of proposals. `SelectiveSearch` is surprisingly effective despite being a fully hand-crafted method (no machine learning involved). When considering less than 10^3 proposals, `MCG`, `Endres2010`, and `CPMC` provide strong results.

Overall, we can group the methods into two groups: well localized methods that gradually lose recall as the IoU threshold increases and methods that only provide approximate bounding box locations, so their recall drops dramatically. All baseline methods, as well as `Bing`, `Rahtu2011`, and `EdgeBoxes` fall into the latter category. `Bing` in particular, while providing high repeatability, only provides high recall at IoU $= 0.5$ and drops dramatically when requiring higher overlap.

When inspecting figure 4 from left to right, one notices that with only few proposal windows our baselines provide low recall compared to non-baseline methods (figure 4a). However as the number of proposals increases baselines such as `Gaussian` or `Sliding-Window` become competitive (figure 4b, IoU > 0.7). Eventually for enough windows even simple random baselines such as `Uniform` become competitive (figure 4c). In relative gain, detection proposal methods have most to offer for low numbers of windows.

Figure 6 presents the results over the ImageNet validation dataset. Compared to Pascal VOC 2007 test set, it includes $10\times$ ground truth classes and $4\times$ images. Somewhat surprisingly the results are almost identical to the ones in 4b. A more detailed inspection reveals that, by design, the statistics of ImageNet match the ones of Pascal VOC. In particular the typical image size, and the mean number of object annotation per image (three) is identical in both datasets. This explains why the recall behaviour is similar, and why `Selective-Search`, `EdgeBoxes`, and `MCG` still perform well despite being crafted for Pascal VOC.

Although the curves look very similar, ImageNet covers 200 object categories, many of them unrelated to Pascal VOC. The good news is that having almost identical results in both datasets confirms that these methods do transfer adequately amongst object classes, and thus can be considered as “true objectness” measures. In other words, the experiments of figure 6 indicate that there is no visible over-fitting of the candidate detection methods towards the 20 Pascal VOC categories.

For researchers looking to benchmark their detection proposal method, figure 5a serves as good overall summary. It puts aside the IoU threshold, because detection proposal users can typically estimate their computation constrains (maximum number of proposals) better than the needed detection proposals localization (IoU threshold). If a specific IoU threshold is desired, then summary figure such as 5b or 5c are suitable.

Method	mAP
LM-LLDA	**33.5/34.4**
Objectness	25.0/25.4
CPMC	29.9/30.7
Endres2010	31.2/31.6
Sel.Search	31.7/32.3
Rahtu2011	29.6/30.4
Rand.Prim	30.5/30.9
Bing	21.8/22.4
MCG	**32.4/32.7**
Ranta.2014	30.7/31.3
EdgeBoxes	31.8/32.2
Uniform	16.6/16.9
Gaussian	**27.3/28.0**
Slid.Window	20.7/21.5
Superpixels	11.2/11.3

Table 1: Detection results on Pascal 2007 (left/right mAP is before/after bounding box regression).

5 Using the detection proposals

This section analyses the detection proposals in the context of a DPM object detector, namely the LM-LLDA variant [14]. We use a pre-trained model and different proposals to filter its detections at test time. This experiment does not speed-up the detection, but enable evaluating the effect of proposals on the detection quality. A priori detections might get worse (by losing recall), but can also improve if the detection proposal method discards windows that would otherwise be false positives.

Implementation details We take the raw detections of an LM-LLDA model before non-maximum suppression and filter them with the detection proposals of each method. We keep all detections that overlap more than 0.8 IoU with a candidate; the surviving detections are then non-maxima suppressed. As final step we do bounding box regression, as common for DPM models [12]. Note that with this procedure the detector is not only evaluated inside each proposal window, but also around it.

Results Table 1 shows that using 1 000 detection proposals decreases detection quality compared to sliding window. When we compare these results with figure 4b, we see that methods with high area under the curve (above 69%), also have high mAP. Although Rahtu and the Gaussian baseline have a similar area under the curve as Objectness and Bing, the latter have are lower mAP than the former. We attribute this to higher recall in the high precision area (IoU ≥ 0.7). These results show that better localisation of proposals leads to increased detection quality and support the analysis in §4.

The per class recall versus precision plots (in supplementary material) can be grouped into three cases: 1) the best proposal methods do not gain or lose much; 2) proposals sometimes clearly hurt performance (bicycle, bottle, car, chair, motorbike, person, potted plant); 3) proposals improve performance (aeroplane, cat, dining table, dog, sofa). In the case of (2) we see reduced recall but also reduced precision, probably because bad localization decreases scores of strong detections.

6 Conclusion

We evaluated ten detection proposal methods, among which SelectiveSearch and EdgeBoxes strike by their consistently good performance both in ground truth recall, reasonable repeatability, and tolerable evaluation speed (see table 2). If fast proposals are required properly tuned EdgeBoxes seem to provide the best compromise in speed versus quality. When requiring less than 10^3 proposals, MCG is the method of choice for high recall if speed is not a concern. MCG also obtains top results regarding pure detection quality (table 1).

Method		Time	Repeatability	Recall	Detection
Objectness[1]	O	3	·	⋆	·
CPMC[4]	C	250	-	⋆⋆	⋆
Endres2010[9]	E	100	-	⋆⋆	⋆⋆
Sel.Search[30]	SS	10	⋆⋆	⋆⋆⋆	⋆⋆
Rahtu2011[24]	R1	3	·	·	⋆
Rand.Prim[22]	RP	1	⋆	⋆	⋆
Bing[6]	B	0.2	⋆⋆⋆	⋆	·
MCG[3]	M	30	⋆	⋆⋆⋆	⋆⋆
Ranta.2014[25]	R4	10	⋆⋆	·	⋆
EdgeBoxes[33]	EB	0.3	⋆⋆	⋆⋆⋆	⋆⋆
Uniform	U	0	·	·	·
Gaussian	G	0	·	·	⋆
SlidingWindow	SW	0	⋆⋆⋆	·	·
Superpixels	SP	1	⋆	·	·

Table 2: Overview of detection proposal methods. Time is in seconds. Repeatability, quality, and detection rankings are provided as rough qualitative overview; "-" indicates no data, "·", "⋆", "⋆⋆", "⋆⋆⋆" indicate progressively better results. See paper's text for details and quantitative evaluations.

The extensive evaluation enables researchers to make more informed decisions when considering to use detection proposals. We hope that the provided code and data will encourage authors of detection proposal methods to provide more complete comparisons to related work. Our study also has revealed that most method suffer from low repeatability due to unstable superpixels, even for the slightest of image perturbation. We foresee room for improvement by using more robust superpixel (or boundary estimation) methods. Finally, our ImageNet experiments have validated that most methods do indeed generalise to different unseen categories (beyond Pascal VOC), and as such can be considered true "objectness" methods.

In future work we plan to study the issue of missing recall in more detail and do further experiments regarding the relation between detection proposals and object detection.

References

[1] B. Alexe, T. Deselaers, and V. Ferrari. What is an object? In *CVPR*, June 2010.

[2] B. Alexe, T. Deselares, and V. Ferrari. Measuring the objectness of image windows. *PAMI*, 2012.

[3] Pablo Arbelaez, J. Pont-Tuset, Jon Barron, F. Marqués, and Jitendra Malik. Multiscale combinatorial grouping. In *CVPR*, 2014.

[4] J. Carreira and C. Sminchisescu. Constrained parametric min-cuts for automatic object segmentation. In *CVPR*, 2010.

[5] J. Carreira and C. Sminchisescu. Cpmc: Automatic object segmentation using constrained parametric min-cuts. *PAMI*, 2012.

[6] Ming-Ming Cheng, Ziming Zhang, Wen-Yan Lin, and Philip H. S. Torr. BING: Binarized normed gradients for objectness estimation at 300fps. In *CVPR*, 2014.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.

[8] P. Dollár, R. Appel, and W. Kienzle. Crosstalk cascades for frame-rate pedestrian detection. In *ECCV*, 2012.

[9] Ian Endres and Derek Hoiem. Category independent object proposals. In *ECCV*, 2010.

[10] Ian Endres and Derek Hoiem. Category-independent object proposals with diverse ranking. In *PAMI*, 2014.

[11] Mark Everingham, S. M. Ali Eslami, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge – a retrospective. *IJCV*, 2014.

[12] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 2010.

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

[14] R. Girshick and J. Malik. Training deformable part models with decorrelated features. In *ICCV*, 2013.

[15] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[16] C. Gu, J. Lim, P. Arbelaez, and J. Malik. Recognition using regions. In *CVPR*, 2009.

[17] C. Gu, P. Arbelaez, Y. Lin, K. Yu, and J. Malik. Multi-component models for object detection. In *ECCV*, 2012.

[18] H. Harzallah, F. Jurie, and C. Schmid. Combining efficient object localization and image classification. In *ICCV*, 2009.

[19] Derek Hoiem, Yodsawalai Chodpathumwan, and Qieyun Dai. Diagnosing Error in Object Detectors. In *ECCV*, 2012.

[20] Ahmad Humayun, Fuxin Li, and James M. Rehg. Rigor: Recycling inference in graph cuts for generating object regions. In *CVPR*, 2014.

[21] Hongwen Kang, Alyosha Efros, Takeo Kanade, and Martial Hebert. Data-driven objectness. *PAMI*, 2014.

[22] Santiago Manén, Matthieu Guillaumin, and Luc Van Gool. Prime object proposals with randomized prim's algorithm. In *ICCV*, 2013.

[23] Krystian Mikolajczyk, Tinne Tuytelaars, Cordelia Schmid, Andrew Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *IJCV*, 2005.

[24] E. Rahtu, J. Kannala, and M. Blaschko. Learning a category independent object detection cascade. In *ICCV*, 2011.

[25] P. Rantalankila, J. Kannala, and E. Rahtu. Generating object segmentation proposals using global and local search. In *CVPR*, 2014.

[26] Antonio Torralba, Kevin P. Murphy, and William T. Freeman. Sharing visual features for multiclass and multiview object detection. *PAMI*, 2007.

[27] Tinne Tuytelaars. Dense interest points. In *CVPR*, 2010.

[28] Tinne Tuytelaars and Krystian Mikolajczyk. Local invariant feature detectors: a survey. *Foundations and Trends in Computer Graphics and Vision*, 2008.

[29] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. *IJCV*, 2013.

[30] Koen van de Sande, Jasper Uijlings, Theo Gevers, and Arnold Smeulders. Segmentation as selective search for object recognition. In *ICCV*, 2011.

[31] P. Viola and M. Jones. Robust real-time face detection. In *IJCV*, 2004.

[32] Xiaoyu Wang, Ming Yang, Shenghuo Zhu, and Yuanqing Lin. Regionlets for generic object detection. In *ICCV*, 2013.

[33] C. Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, 2014.

How good are detection proposals, really? Supplementary material

Jan Hosang
http://mpi-inf.mpg.de/~jhosang
Rodrigo Benenson
http://mpi-inf.mpg.de/~benenson
Bernt Schiele
http://mpi-inf.mpg.de/~schiele

MPI Informatics
Saarbrücken, Germany

1 Method overview

Table 1 presents an overview of all eleven methods listed in §2 of the paper. For three of the methods, no code is available, so we evaluate the remaining eight. The table also shows the four baseline methods.

Some of the methods have a parameter to ask for a certain number of proposals and either return roughly that number of candidates (denoted by "R") or return significantly less (denoted by "LR"). Methods without a parameter to directly ask for a number of proposals return a number of proposals that depends on the image content and size. These proposals are either sorted by likelihood of containing an object (denoted by "Fs") or in an arbitrary order (denoted by "F").

2 Controlling the average number of detection proposals

This section gives the details on how exactly we run the different detection proposal methods and how we obtain the desired number of proposals.

`gPbUCM` has no code available.

`Objectness` scores every detection proposal, so for selecting a subset, we use the highest scored ones.

`CPMC` has no parameter that directly controls the number of produced proposals, but provides a score. We use the highest scoring proposals if we need less than were produced.

`Endres2010` has no parameter to control the number of proposals. The list is, however, presorted so we use the first proposals to select subsets.

`Selective Search` returns a randomized priority, which can be used to select a subset of proposals. Low priorities are selected first.

	Method	Low level	Output type	Score	#windows	Time	Repeatability	Quality	Detection
Pb	gPbUCM	Own	Sg	No	F	-	-	-	-
O	Objectness	sp	Bx	Yes	R	3	·	⋆	·
C	CPMC	Own	Sg	Yes	Fs	250	-	⋆⋆	⋆
E	Endres2010	sp	Sg	Yes	Fs	100	-	⋆⋆	⋆⋆
SS	SelectiveSearch	sp	Sg	No	F	10	⋆⋆	⋆⋆⋆	⋆⋆
R1	Rahtu2011	sp	Sg	Yes	R	3	·	·	⋆
RP	RandomizedPrim	sp	Sg	No	LR	1	⋆	⋆	⋆
B	Bing	Own	Bx	Yes	·Fs	0.2	⋆⋆⋆	⋆	·
M	MCG	C⋆	Sg	Yes	R	30	⋆	⋆⋆⋆	⋆⋆
R4	Rantalankila2014	C+sp	Sg	No	F	10	⋆⋆	·	⋆
R	Rigor	C⋆	Sg	No	LR	-	-	-	-
EB	EdgeBoxes	Own	Bx	Yes	Fs	0.3	⋆⋆	⋆⋆⋆	⋆⋆
U	Uniform	∅	Bx	No	R	0	·	·	·
G	Gaussian	∅	Bx	No	R	0	·	·	⋆
SW	SlidingWindow	∅	Bx	No	R	0	⋆⋆⋆	·	·
SP	Superpixels	Own	Sg	No	F	1	⋆	·	·

Table 1: Comparison of different detection proposal methods.
"sp" indicates "super pixels" (of diverse type), C⋆ indicates variants of CPMC, and ∅ indicates "no low-level component".
Sg indicates that the method outputs segments, Bx that it outputs only bounding boxes. F indicates "fix set", Fs "fix sorted set", R "as requested", and LR "less than requested".
Time is indicated in seconds.
Repeatability, quality, and detection rankings are provided as very rough, subjective, guidelines; "-" indicates no data, "·", "⋆", "⋆⋆", "⋆⋆⋆" indicate progressively better results.

Rahtu2011 provides a score that is not monotonically decreasing in the order of the returned proposals. Using the first n, instead of the best n proposals resulted in better results.

RandomizedPrim's provides no scores, since the output is the result of a sampling procedure, that takes learned probabilities into account. After discussion with the authors, we request 20 000 proposals for the quality experiments and 5 000 proposals for the repeatability experiments and filter by using those, that are returned first.

Bing provide a score and we use the highest scoring detection proposals.

MCG We use the first n candidates. Sorting by the returned score worsens results.

Rantalankila2014 has no parameter that controls the number of proposal directly. We run the method with both slic and Felsenzwalb segmentation and control the number of proposals by adapting the parameter `gc_branches`. The parameter settings for the different number of proposals were determined experimentally by running the method with different parameters over 50 images and averaging the number of proposals we get per image. If we need to filter we use the first proposals.

Rigor No code available at time of writing.

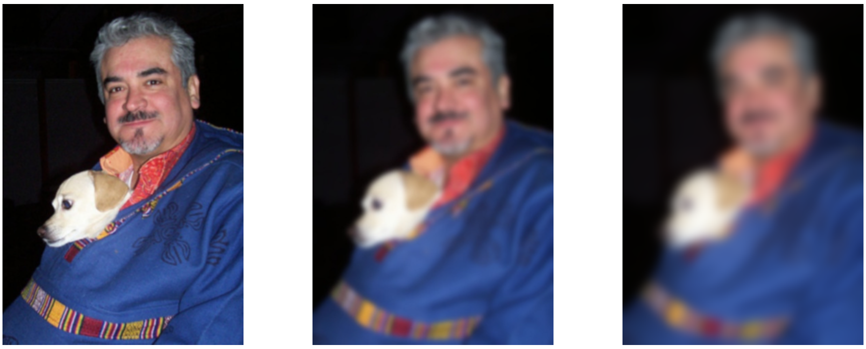

(a) $\sigma = 1$ (b) $\sigma = 4$ (c) $\sigma = 8$

Figure 1: Image blurring examples for different values of σ.

`EdgeBoxes` scores all candidates. We use highest scoring candidates first.

`Uniform` proposals are sampled independently, so we can sample 10 000 proposals and use the first ones if we need a subset.

`Gaussian` same as for Uniform.

`SlidingWindow` always gives at most as many proposals as we ask for, returns no scores, and the samples are not independent. So we rerun the method and ask for an increasing amount of proposals to produce the quality curve.

`Superpixels` runs the Felsenzwalb segmentation method with 4 different parameters, like Randomized Prims does, and returns the bounding boxes of every superpixel. If we need to filter, we use the first proposals. This could, on principle, hurt the repeatability performance, but on average we get 771 proposals on the Pascal test set, so this is not an issue.

3 Perturbations for repeatability

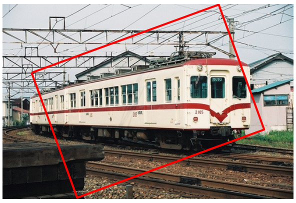
In this section, we explain the details of the image perturbations and give examples.

3.1 Scale

We uniformly sample the scale factor in scale space, from one octave down to one octave up, i.e. 0.5 to 2. Do get an idea of the drop around the identity transform, we add the scaling factors 0.9, 0.95, 0.99, 1.01, 1.05, and 1.1. The scaling operations have been done with the default parameter setting of Matlab's imresize, which means upscaling with bicubic interpolation and downscaling with antialiasing.

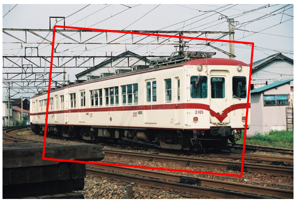
3.2 Blur

We blur the images with a Gaussian kernel with different standard deviations σ. In detail, we construct a Gaussian kernel of size $20 \cdot \sigma$ with the Matlab function fspecial. We use imfilter

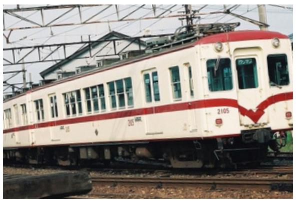


(a) The biggest rectangle with the same aspect ratio as the original image, that can fit into the image with 20 degrees rotation.

(b) The resulting crop from 2a.



(c) Example rotation with of -5 degrees.



(d) The resulting crop of 2c.

Figure 2: Examples of how the rotation transformation is defined.

with symmetric image padding to get an output image of same size. Figure 1 shows some examples.

3.3 Rotation

We rotate the image in 5 degree steps between -20 and 20 degrees. To avoid both 1) padding the image with fake image content and 2) having too many detection proposals in areas that are cropped by the transformation, we do the following: We first construct the biggest bounding box with the same aspect ratio as the original image that can fit into the original image with the most extreme setting of rotation. Figure 2a shows such a rectangle on an example image, while the resulting cropped can be seen in figure 2b. This defeats problem 1). To limit problem 2), we use the size of the previous crop for all rotations, even if there is enough content to make the crop bigger. See figure 2c and 2d for an example.

3.4 Illumination

To synthetically change the illumination of the images we changed the brightness channel in the HSB color space. We do that with the imagemagick library. We chose the extremes of the transformation so that we observe some over and under saturation, as can be seen in figure 3.

3.5 JPEG artefacts

To create JPEG artefacts we write the image to disk with the Matlab function imwrite, which supports JPEG compression and a quality parameter. We try very different quality parame-

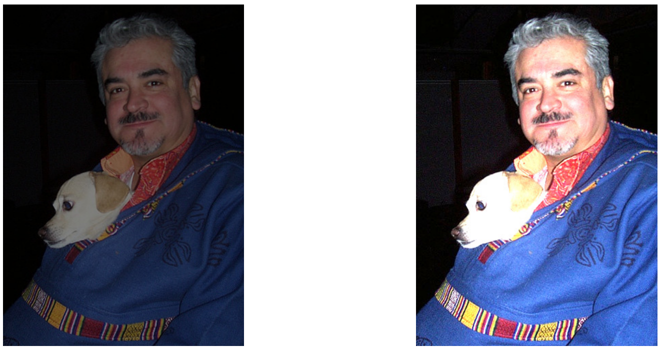

(a) Illumination change to 50%.
(b) Illumination change to 150%.

Figure 3: Extreme examples for illumination changes.

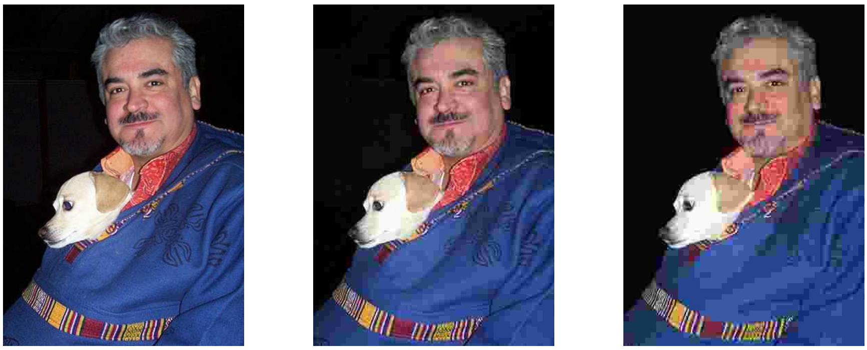

(a) 50% quality
(b) 10% quality
(c) 5% quality

Figure 4: JPEG compression examples.

ters including very low ones and and the highest setting, 100%. We also include a lossless parameter (no image berturbation) setting for comparison.

Figure 4 shows the low end of the quality spectrum, as only that is actually easily visible for humans. When doing a pixel wise comparison between a JPEG compressed image and the original, we see differences on the entire image, even for the 100% quality setting.

4 Repeatability results

We omitted the repeatability plot of the blurring perturbation from the paper due to space constrains. It can be found in figure 5e. For completeness we repeat all other repeatability results in figure 5.

5 Proposal recall

5.1 Pascal VOC 2007

In the paper we analyse the recall as a function of how well the detection proposals have to be localised. Another interesting view on the behaviour of the methods is to plot the recall as a function of the number of proposals we use. Doing this, we are faced with the problem, that the IoU threshold has to be removed from the plot. One solution is to plot the recall for a fixed IoU threshold (figure 6e and 6f). An alternative is to plot the area under the "recall versus IoU thershold curve", i.e. the area under the curves in figure 6a, 6b, and 6c. You can see the area under those curves as the first number in the legend. In figure 6d, you see the area under the curves as the function of the number of proposals.

Note how the three different curves (figure 6d, 6e, and 6f) favour different methods. Picking one of the curves or showing only part of a curve does not tell the full story and can be misleading.

5.2 ImageNet

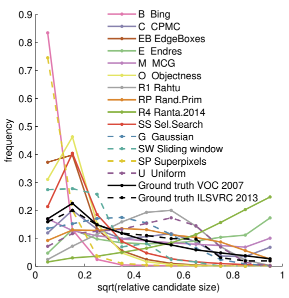
All methods were trained on Pascal and also tuned towards the test set of Pascal it is unclear how they perform on a different dataset. Thus, we also evaluate the recall on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC2013) detection task, which has 200 classes instead of 20 on Pascal. Albeit the difference in size and number of classes, it has to be noted that the distribution of annotation sizes (figure 5a), the distribution of image sizes, and the average number of annotations per image is quite similar.

We do the experiments on the ILSVRC2013 validation set (as the test set annotations are not available). The ILSVRC detection evaluation protocol blacklists some images for some classes, because of too much ambiguity in the annotations. We follow this procedure and do not count those annotations in the recall computation. This does not have a negative impact on the curves because they ignore false positives.

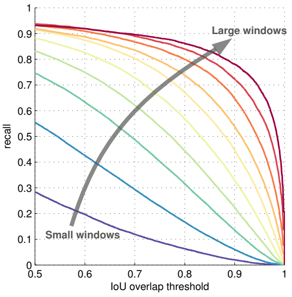
Figure 7 shows the recall results of on the ILSVRC2013 detection validation set.

6 Detection

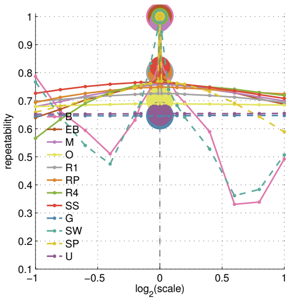
In the paper, we only report the mean average precision (mAP), i.e. results average over all classes. As mentioned in the paper we see different behaviour for the classes, so it is worth supplying all recall precision curves. See figures 8, 9, 10, and 11.



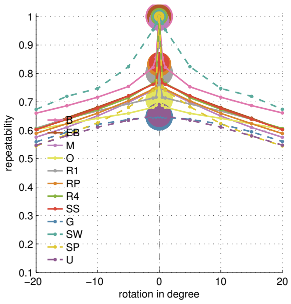
(a) Histogram of proposal windows sizes for different methods on Pascal VOC 2007 test.



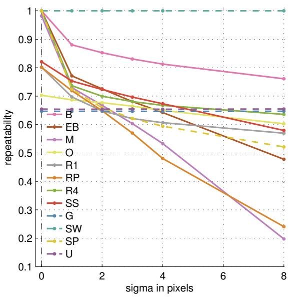
(b) Example of recall fluctuation for different size group (blur perturbation). Each group corresponds to one point in figure 5a.



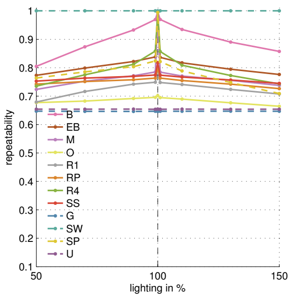
(c) Repeatability under scale changes. x-axis in log scale from scale half to double of the original image size.



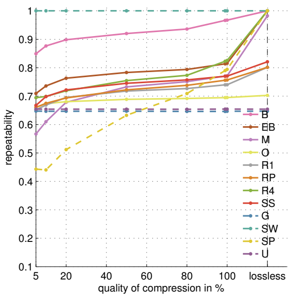
(d) Repeatability under rotation. x-axis shows rotations from −20 to 20 degrees.



(e) Repeatability under blurring. x-axis shows the gaussian kernel standard deviation from 0 to 8 pixels.

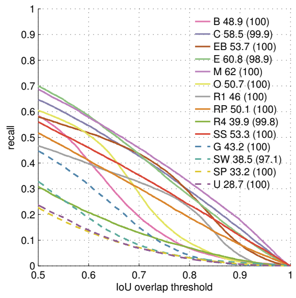


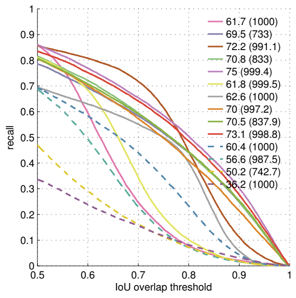
(f) Repeatability under illumination. x-axis shows the changes from dark (50), to neutral (100), to overexposed image (150).

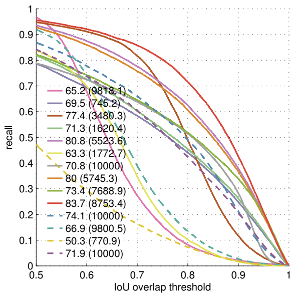


(g) Repeatability under JPEG artefacts. x-axis shows the quality preservation from 100 % (no change) to 5% (obvious artefacts).

Figure 5: Repeatability results.



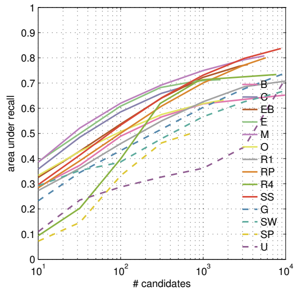


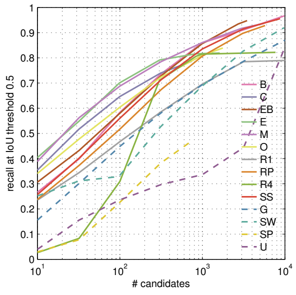


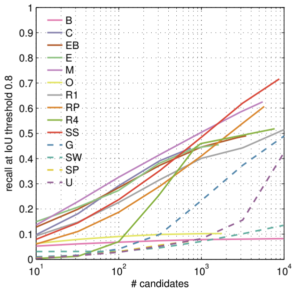
(a) Recall versus IoU threshold, for 100 proposals per image. Curves are labelled with number of proposals/area under recall curve.

(b) Recall versus IoU threshold, for 1 000 proposals per image.

(c) Recall versus IoU threshold, for 10 000 proposals per image.





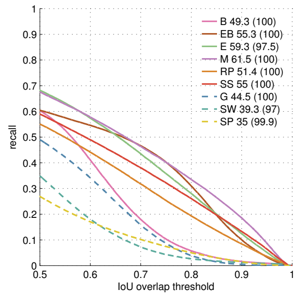


(d) Area under "recall versus IoU threshold" curves, for varying number proposed windows.

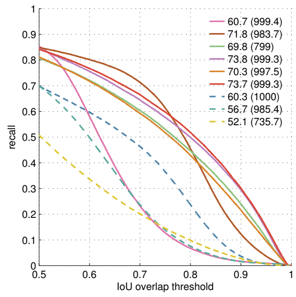
(e) Recall at IoU above 0.5 versus number of proposed windows.

(f) Recall at IoU above 0.8 versus number of proposed windows.

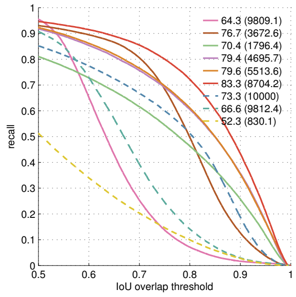
Figure 6: Proposals quality over Pascal VOC 2007 test set. On recall versus IoU threshold curves, number indicates area under the curve, and number in parentheses obtained average number of windows per image.



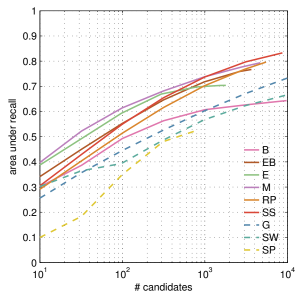
(a) Recall versus IoU threshold, for 100 proposals per image.



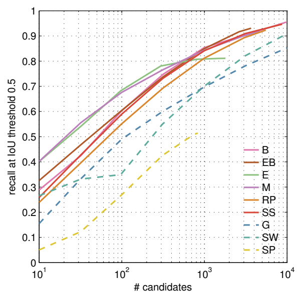
(b) Recall versus IoU threshold, for 1 000 proposals per image.



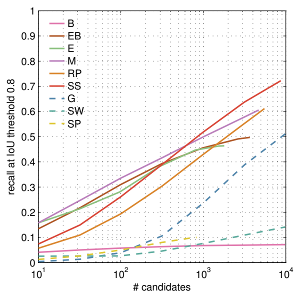
(c) Recall versus IoU threshold, for 10 000 proposals per image.



(d) Area under the "recall versus IoU threshold" curves, for varying number proposed windows.



(e) Recall at IoU above 0.5 versus number of proposed windows.



(f) Recall at IoU above 0.8 versus number of proposed windows.

Figure 7: Proposals quality over ImageNet 2013 validation set. On recall versus IoU threshold curves, number indicates area under the curve, and number in parentheses obtained average number of proposals per image.

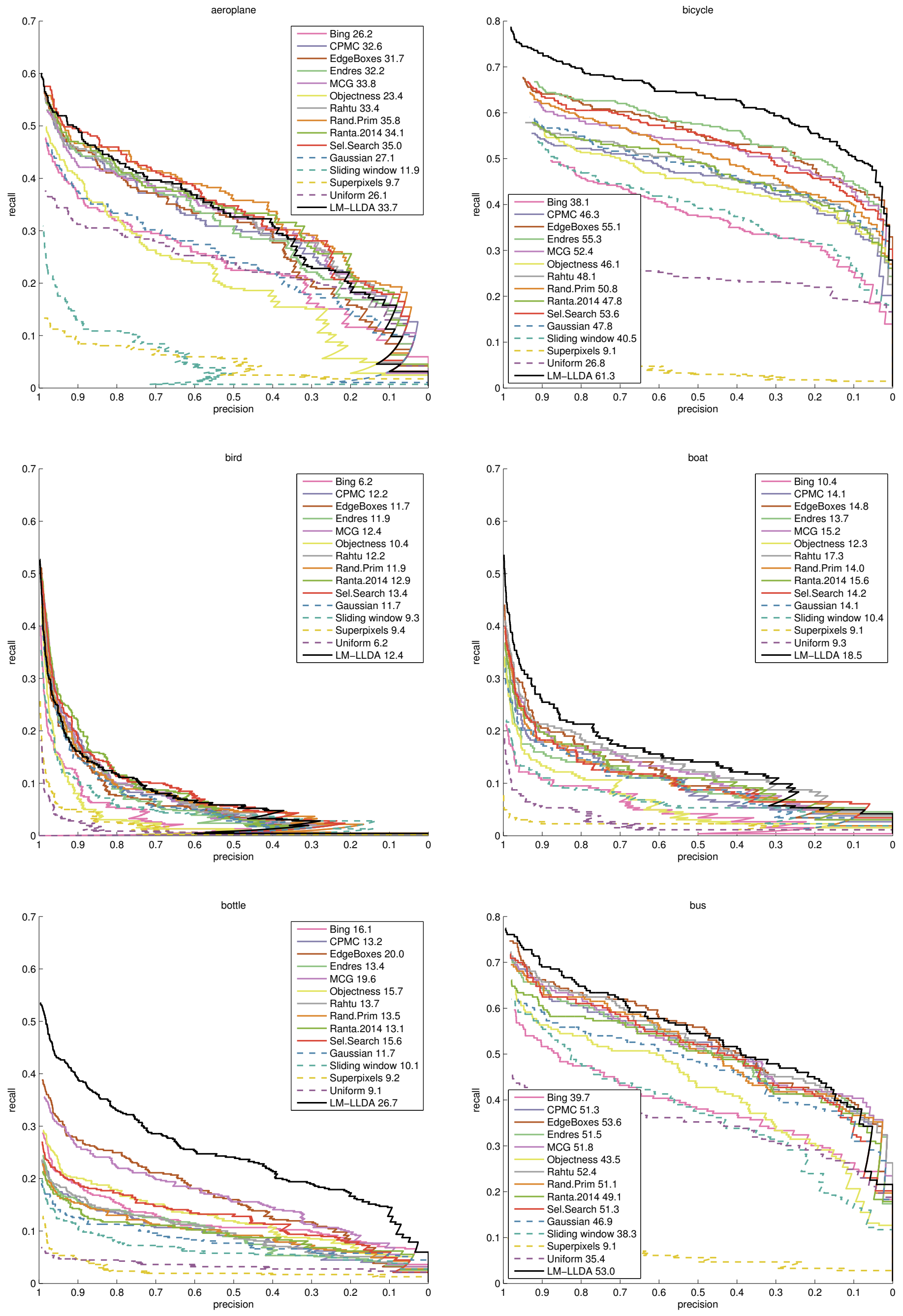


Figure 8: Recall-precision curves for Pascal VOC 2007 using different proposal methods at test time.

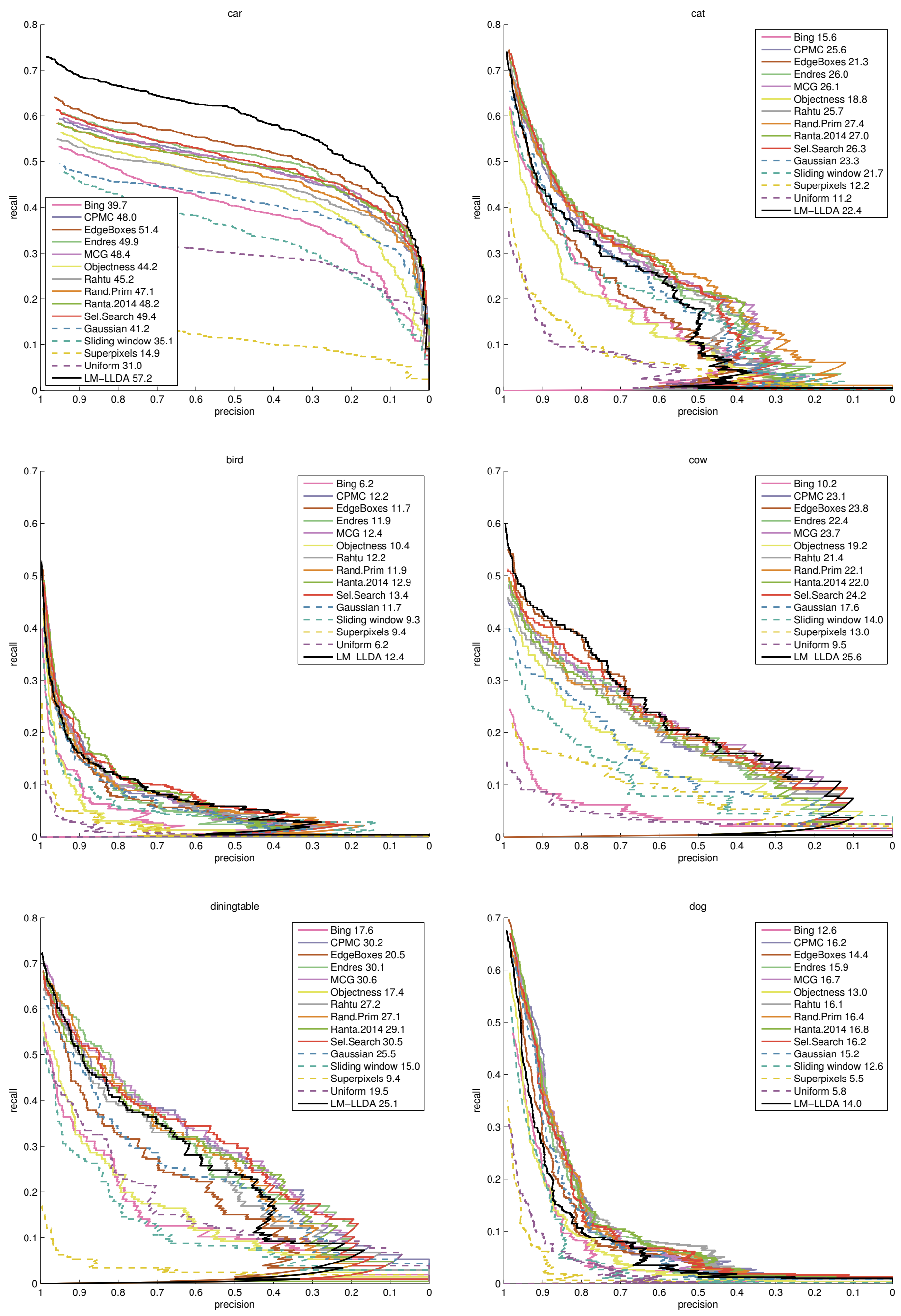


Figure 9: Recall-precision curves for Pascal VOC 2007 using different proposal methods at test time.

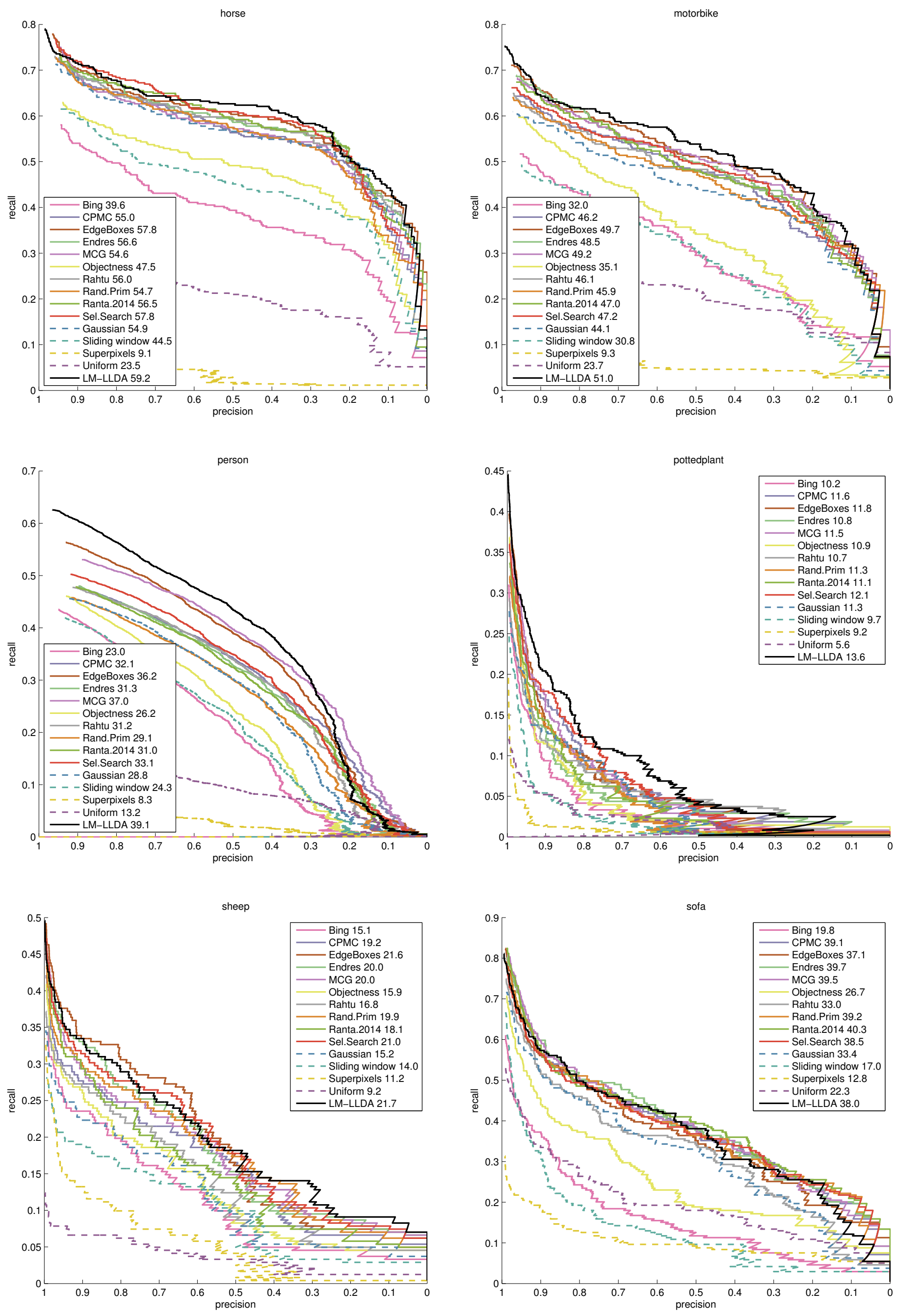


Figure 10: Recall-precision curves for Pascal VOC 2007 using different proposal methods at test time.

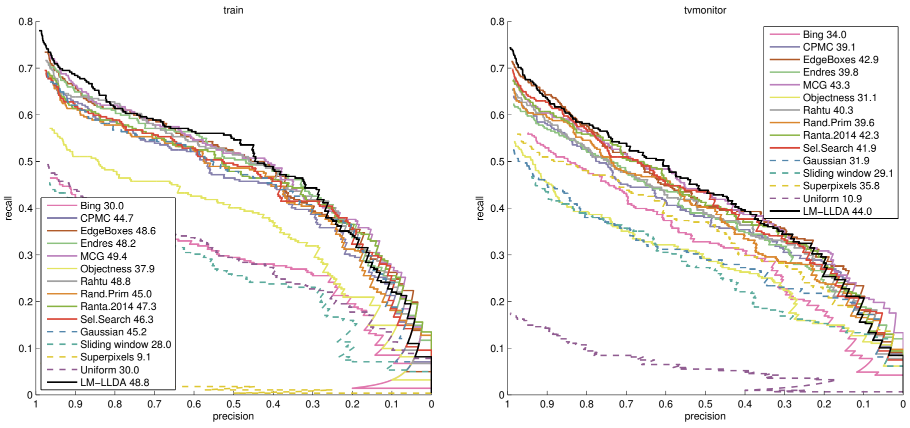


Figure 11: Recall-precision curves for Pascal VOC 2007 using different proposal methods at test time.