Object Proposal by Multi-branch Hierarchical Segmentation

Chaoyang Wang
Shanghai Jiao Tong University
wangchaoyang@sjtu.edu.cn

Long Zhao
Tongji University
9012garyzhao@tongji.edu.cn

Shuang Liang*
Tongji University
shuangliang@tongji.edu.cn

Liqing Zhang
Shanghai Jiao Tong University
zhang-lq@cs.sjtu.edu.cn

Jinyuan Jia
Tongji University
jyjia@tongji.edu.cn

Yichen Wei
Microsoft Research
yichenw@microsoft.com

Abstract

Hierarchical segmentation based object proposal methods have become an important step in modern object detection paradigm. However, standard single-way hierarchical methods are fundamentally flawed in that the errors in early steps cannot be corrected and accumulate. In this work, we propose a novel multi-branch hierarchical segmentation approach that alleviates such problems by learning multiple merging strategies in each step in a complementary manner, such that errors in one merging strategy could be corrected by the others. Our approach achieves the state-of-the-art performance for both object proposal and object detection tasks, comparing to previous object proposal methods.

1. Introduction

The goal of generic object proposal generation is to find all candidate regions that may contain objects in an image [1]. A generic object proposal algorithm has the following requirements: it should be able to capture objects of all scales, has small bias towards object class, and most importantly: achieve high detection recall with as few proposals as possible. The problem has received intensive interests in recent years, as it serves as an effective preprocessing for other high-level computer vision tasks such as object detection. Recent works [14, 16, 15] demonstrate that the proposal algorithm can indeed significantly speed up the object detection task and improves its performance.

Generic object proposal is first addressed within traditional sliding window object detection framework [1]. Methods such as [1, 22, 9, 7, 28] use features like saliency, image gradient and contour information to measure the objectness of a given sliding window.

Different from the above, methods such as [19, 6, 21, 26, 5, 23] address the problem from the image segmentation perspective. Starting from an initial oversegmentation, object proposals are modeled as a composition of neighboring segments. This segment composition space is exponential, so the principle of all segment-based methods is to use efficient strategy such as automatic foreground-background segmentation [19, 6], randomized prim sampling [21], hierarchical image segmentation [26], combinatorial grouping [5], or a combination of above [23] to search the segment composition space.

In this work, we develop our approach based on hierarchical image segmentation. As proposed in [26], using segments from hierarchical segmentation as proposal regions handles multiple scales by natural. Moreover, organizing object proposals with a hierarchical structure coincides with the intuition that the semantic meanings of real world objects are hierarchical.

Hierarchical segmentation divides an image into a hierarchy of regions [20]. Each region is visually or semantically homogeneous, and regions in lower level of the hierarchy form subparts of regions in higher levels. It is most commonly implemented as a bottom-up greedy merging process [4, 12, 23, 26, 24]. Starting from an initial oversegmentation, the image segmentation hierarchy is generated by iteratively merging neighboring segments in a greedy manner: a merging strategy gives a score to each pair of adjacent segments and the pair with the highest score is merged. This greedy merging process continues until either the whole image is merged into one segment [26, 4] or the maximum merging score is below some threshold [23, 24, 12].

The merging strategy has been defined in various ways. The method in [4, 3, 5] gives merge scores using global contour detection result; the methods in [12, 26, 23] heuristically define merging strategies based on appearance similarity and segment's size; the methods in [24, 27] trains a cascade of linear classifiers and uses classifiers' decision value as merge score.

*Corresponding author.

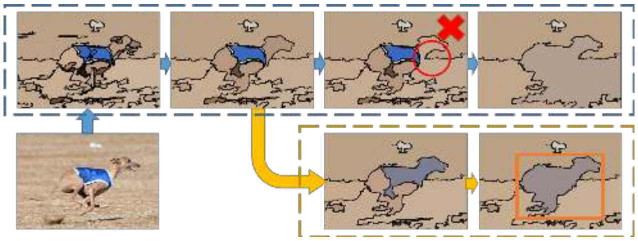

Figure 1. Illustration of hierarchical segmentation with branching. Bottom left: input image. Top: greedy merging process using only one merging strategy. The red circle indicates an incorrect merging: the dog's head is merged to a piece of background and it is lost. Bottom right: following the yellow arrows, by using the second complementary merging strategy, the dog is finally successfully detected.

The performance of bottom-up greedy merging heavily relies on the accuracy of the merging strategy because any mistakes made in early stages are irremediable due to its greedy nature. In the context of object proposal problem, once a part of an object is incorrectly merged with the background, this object has little chance to be found later. However, since most of the real world objects are very complicated, direct using segments from a single hierarchical segmentation achieves low object detection recall. To increase recall, the method in [26] directly combines object proposals from a set of hierarchical segmentations with various merging strategies and initial oversegmentations. The method in [23] combines proposals generated by searching from different foreground seeds.

To search the segment composition space more effectively, we propose our *multi-branched hierarchical segmentation* based on a more general principle: an object consists of subparts with various colors and textures may require a combination of different merging strategies to bring it together. As illustrated in Fig. 1: given an image of a running dog wearing a blue jacket, the most salient boundary in the whole image is between the jacket and the dog's three separate body parts. The first row shows the merging process which uses color and texture similarity as merge score. After merging superpixels into visually homogeneous parts, it incorrectly merges the dog's head with a piece of background (marked by the red circle), and this leaves the subsequent merging no chance to detect the dog. Our solution for this is branching. By starting a new branch with a new merging strategy specifically designed to be complementary to the first merging strategy right before where it went wrong (shown in the bottom row), the dog is successfully detected in the branch.

To make our multi-branched hierarchical segmentation effective, we address two important problems: how to design complementary merging strategies capable of fixing mistakes made by others, and how to organize branching since arbitrary branching causes exponentially large search space. We handle those issues as below:

Learning complementary merging strategies. We model the merging strategies as binary linear classifiers. We propose an automatic learning procedure that trains the complementary merging strategies (classifiers) by altering weights for each training samples in a boosting-like fashion. Different from traditional boosting [13], our goal is not to combine the classifiers into a strong one, but to obtain a set of complementary ones to enrich the chances of finding objects missed in each other. This makes the later classifiers capable of fixing the mistakes made in earlier classifiers.

Multi-staged branching. Using the complementary merging strategies (classifiers), we branch the searching into multiple directions. New branches start only when the classifier's decision scores of all pairs of adjacent segments are below zero. This setting splits one single greedy merging process into multiple stages. Merging strategies in each stages are different.

Since the total number of branches at the last stage is exponential to the stage number, it appears that this multi-staged branching would generate too many proposals. But as shown in Sec. 3, if we use a small branch degree (2 in our experiment) and a large merging pace (merges 50% pairs of segments) in each stage, the proposal number can be kept relatively small.

The segmentation method in [24] also organizes the greedy merging process in cascaded stages with learned merging strategies, but based on different principles. Compared to their work, we emphasize on improving object detection recall by branching with complementary merging strategies, rather than high boundary recall with single sequence of merging.

Extensive comparison to previous object proposal methods indicates that our approach achieves the state-of-the-art results in terms of object proposal evaluation protocols. Moreover, in order to investigate how well these methods perform for real-world object detection tasks, we test all compared object proposal methods using the state-of-the-art R-CNN detector [14] on PASCAL VOC2007 [10]. To the best of our knowledge, we are the first to give this end to end comparison. [1]

2. Related Work

The existing object proposal algorithms generally fall into two categories: scoring-based and segment-based. We review the two categories separately in the below. A more comprehensive survey is provided in [18].

The pioneer work in [1, 2] firstly addresses the problem of measuring the generic objectness scores of image

[1] The contemporary work [17] also perform such an object proposal evaluation using the R-CNN detector.

windows. It adopts a variety of appearance and geometry properties contained in an image window to measure how likely there exists an object. It is improved in [22, 9] by using more complicated features and learning methods. Recent works in [7] and [28] score image windows by simple concepts using only image gradients and contours [8] and they show very fast computational speed compared to other methods.

On the other hand, segment-based methods such as [19, 6, 21, 26, 5, 23] address the generic object proposal from a different angle. Starting from an initial oversegmentation, their principle is to use efficient strategy such as automatic foreground-background segmentation [19, 6], randomized prim sampling [21], hierarchical image segmentation [26], combinatorial grouping [5], or a combination of above [23] to search the segment composition space. The approach in [26] has been proven effective by recent object detection works. Due to its high recall and reasonable computational speed, many state-of-the-art object detection algorithms [14, 16] use its proposals as input object candidates.

Our work is mostly related to [26], as both use hierarchical image segmentation and multiple merging strategies. Our approach differs in that it is built on two more general principles: multi-staged branching and automatic learning of complementary merging strategies. As shown in experiments, our method is more effective as it finds a smaller number of proposal with higher qualities.

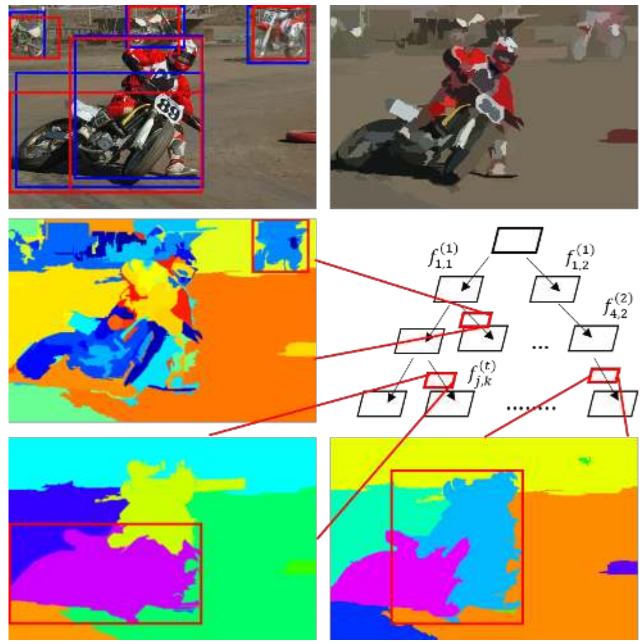

Figure 2. Illustration of our approach. Top left: ground truth bounding box annotation (blue), our proposal windows with highest IoU score (red). Top right: initial oversegmentation rendered with mean color. Mid right: the tree structure of multi-branch hierarchical image segmentation. Bottom & mid left: three objects successfully detected at different branches.

3. Multi-branch Hierarchical Segmentation

3.1. Greedy merging

Hierarchical segmentation is commonly implemented as a bottom-up greedy merging process [4, 12, 23, 26, 12]. Starting from an initial oversegmentation, it evaluates the merging score for each pair of neighboring segments with some merging strategy. Then in each merging iteration, the pair of segments with the highest score is merged together. This process carries on until either the whole image is merged into one segment or the maximum score of the remaining pairs are below some threshold.

3.2. Cascaded multi-branch greedy merging

In order to enrich the search capability for complex objects, we propose to try different merging strategies throughout the greedy merging process, which we call *branching*. This approach turns the original hierarchical image segmentation's sequential evolution structure into a tree-like structure. As illustrated in Fig. 2, objects will have a better chance to be detected in one of those tree branches.

For each branch, we model the merging strategy as a binary classifier by using its decision value as merge score. For the branches starting from the same parent branch, their merging strategies are trained to be complementary in a sequential manner, which makes each of them capable of fixing some mistakes made by the others. We describe how our merging strategies are learned in Sec. 3.3

We organize branching by splitting the greedy merging process into several cascaded stages. Algorithm 1 illustrates our algorithm. During each stage, current branches progress until no pair of adjacent regions are classified as positive, which pronounces the end of that stage. Then at the beginning of the next stage, each branch splits into K new branches with K complementary merging strategies. This process carries on until the last stage. The remaining segments are then merged greedily until the whole image is merged as one segment.

One may worry that such a multi-branch tree structure will bring exponentially large number of object proposals. Indeed, a T-staged K-branch greedy merging can generate a total of K^T different hierarchical segmentation results. Despite that the number of segmentations is exponential to the number of the stages, the number of proposal number can be kept relatively low if we use suitable parameters as shown below:

Suppose each stage reduces current segment number by a portion of λ, a T-staged K-branch greedy merging roughly produces $\sum_{t=1}^{T} K^t N(1-\lambda)^{t-1}\lambda$ proposal windows, where N is the superpixel number for the initial over-

Algorithm 1 Multi-branch hierarchical image segmentation algorithm for object proposal generation.

Input: stage number T, branch degree K
Input: initial oversegmentation $S_0 = \{s_1, ..., s_N\}$
Input: classifiers for each branch $f_{j,k}^{(t)}$
Input: thresholds for each branch $b_{j,k}^{(t)}$
Initialize: proposal region set: $\mathcal{R} = S_0$, segmentation queue: $Q = \{S_0\}$.
for $t = 1$ to T **do**
    $branch\_num \leftarrow |Q|$
    for $j = 1$ to $branch\_num$ **do**
        for $k = 1$ to K **do**
            $(P, S) \leftarrow$ greedy merging S_j via $f_{j,k}^{(t)}$, $b_{j,k}^{(t)}$
            Add segmentation S to the back of queue Q
            $\mathcal{R} \leftarrow \mathcal{R} \cup P$
        end for
        Remove S_j from Q.
    end for
end for
return bounding boxes of each regions in $\mathcal{R}$

segmentation. If we set $K = 2$, and $\lambda = 0.5$, the result proposal number is just TN, which is linear to the stage number. Therefore, by setting a reasonable pace (merging speed λ) and branch degree (2 in experiment) for each stage, our multi-branch hierarchical image segmentation algorithm can explore more different segmentation results while keeping the number of overall proposal windows not too large.

We propose to control the pace of each stage by setting a target miss rate τ for each classifier f_k by searching a bias threshold b_k. We empirically set this target miss rate τ as 0.7. This means roughly 30% of the regions which belongs to some objects will be merged in each stage. In test time, we found that about 30% to 60% of regions are merged in each stage, and for an image with 400 initial segments, a 4-staged 2-branched greedy merging produces roughly 900 proposal windows after removing duplicates.

3.3. Complementary merging strategy learning

Our key motivation is, in order to detect real-world objects, a combination of different merging strategies is necessary. We propose to try different merging strategies throughout the greedy merging process. To make this miss-and-fix style framework effective, merging strategies for each branch should be complementary. This means each of them are supposed to be able to fix some mistakes made by the others. To achieve this requirement, we propose an automatic learning procedure, which models merging strategies as binary linear classifiers, and train them in sequence.

We use PASCAL VOC2007's segmentation dataset [10] for training. For each image, a ground truth segmentation map is given, which splits the image into three types of regions: the objects, the background and the void (or unlabelled) regions.

For clarity, we first describe how to train merging strategies for branches starting from the same parent branch. Then we show how to train merging strategies for multiple stages. The whole pipeline is summarized in Algorithm 2.

Training for branches starting from the same parent branch. From an initial input image segmentation produced by the parent branch, we classify each pair of neighboring segments in the initial segmentation into three categories. It is *positive* if both segments belong to the same object in ground truth. It is *negative* if the two segments belong to different objects, or one of them belongs to the background. Otherwise, it is *unknown*. We use the positive and negative pairs as our training samples.

We use linear SVM with weighted loss [11] to train a set of linear classifiers $\{f_k(\cdot)\}$ sequentially. The loss weights $\alpha_i^{(k)}$ for each samples are defined following the two intuitions below:

Diversify classifier's preference. Weighted loss is used to simulate different data's prior distributions. Similar as Boosting [13], we increase the loss weight for each training sample if it's wrongly classified by previous classifier. This makes the classifiers complementary to each other.

Balance training set. Given an image with hundreds of segments, positive samples largely outnumber negative samples since most pairs of neighboring segments are waiting to be merged. Therefore, we need to properly increase the loss weight for those negative samples in order to keep a reasonably small false positive rate. This is important to reduce the effect of early mistakes, which are irremediable in greedy merging.

The learning procedure works as follow:

1. Suppose some weight $p_i^{(k)}$ with initial value 1 is already assigned to each training sample x_i. Instead of directly using $p_i^{(k)}$ as weights in the loss function, the loss weight $\alpha_i^{(k)}$ for each sample is computed by balancing positive and negative samples' total weights:

$$\alpha_i^{(k)} = \frac{N}{2}\left(\frac{\iota(y_i)}{P_{pos}^{(k)}} + \frac{\iota(1-y_i)}{P_{neg}^{(k)}}\right)p_i^{(k)}, \tag{1}$$

   where $\iota(\cdot)$ is the indicator function, which output 1 if input number is positive, and 0 otherwise; N is the training set size, and $P_{pos}^{(k)}$, $P_{neg}^{(k)}$ are the sum of positive/negative training samples' weight $p_i^{(k)}$.

2. Train the k^{th} linear classifier $f_k(\cdot)$ with loss weights $\alpha_i^{(k)}$, and evaluate it over the training set: $\hat{y}_i^{(k)} = f_k(x_i)$.

3. Increase $p_i^{(k)}$ by β times if the sample x_i is wrongly classified by $f_k(\cdot)$:

$$p_i^{(k+1)} = \beta\ \iota(1 - y_i \hat{y_i}^{(k)}) p_i^{(k)}. \quad (2)$$

4. Return to step 1. to train the next classifier $f_{k+1}(\cdot)$.

We control the degree of diversity by the parameter β. A larger β makes the classifier more focusing on the wrong samples and behaving more differently to its predecessors, since the altered training data distribution emphasizes more on the mistakes made by previous classifiers. However such increase of diversity comes with a price of classification precision reduction. As shown in Fig. 3, if β is too large, the detection recall would slightly decrease. In our experiment, we empirically set $\beta = 2$.

Training for multiple stages. In order to fit the evolution of this multi-branch greedy merging process as closely as possible, our complementary classifiers are trained specifically for each branch in different stages. The thresholds used to control stage pace are searched on a separate validation set (we use the training/validation split provided by PASCAL VOC2007's segmentation dataset). As illustrated in Algorithm 2 , at each stage t, the K^{t-1} sets of complementary linear classifiers are learnt using the previously described learning algorithm. These complementary linear classifiers creates K^t new branches. Then for each such branch, its classifier's threshold is searched from the validation set, and new segmentation result on training/validation images are generated for the next stage.

Algorithm 2 Learning merging strategies for multi-branch hierarchical segmentation.

Input: stage number T, branch degree K,
target miss rate τ
Input: training/validation image sets $\mathcal{I}_{train}$, $\mathcal{I}_{valid}$
Input: ground truth segmentation annotations: $\mathcal{G}$
generate initial segmentations: S^0_{train}, S^0_{valid}
init queue: $Q_{train} \leftarrow \{S^0_{train}\}$
init queue: $Q_{valid} \leftarrow \{S^0_{valid}\}$
for $t = 1$ to T **do**
  for $j = 1$ to K^{t-1} **do**
    $S_{train} \leftarrow Q_{train}$.POP()
    $S_{valid} \leftarrow Q_{valid}$.POP()
    $S_{train} \leftarrow$ collect train samples from $S_{train}, \mathcal{G}$
    $S_{valid} \leftarrow$ collect valid samples from $S_{valid}, \mathcal{G}$
    $\{f^{(t)}_{j,k}\} \leftarrow$ train K complementary classifiers from X_{train}
    for $k = 1$ to K **do** ▷ prepare data for next stage.
      predict on X_{valid} with $f^{(t)}_{j,k}$
      search bias term $b^{(t)}_{j,k}$ s.t. miss rate = τ
      $S^k_{train} \leftarrow$ GREEDY_MERGE($S_{train}, f^{(t)}_{j,k}, b^{(t)}_{j,k}$)
      $S^k_{valid} \leftarrow$ GREEDY_MERGE($S_{valid}, f^{(t)}_{j,k}, b^{(t)}_{j,k}$)
      Q_{train}.PUSH(S^k_{train})
      Q_{valid}.PUSH(S^k_{valid})
    end for
  end for
end for
return complementary classifiers $f^{(t)}_{j,k}$ and thresholds $b^{(t)}_{j,k}$

4. Experiments

4.1. Implementation details

We use the same initial oversegmentations and features as in [26], for their simplicity and efficiency. Moreover, using the same features and oversegmentations enables us to exhibit the improvements caused by our automatic learning and multi-branched hierarchical segmentation.

In details, we utilize the region-based features in [26] to train our classifiers, including color histogram intersection, texture histogram intersection, union region area over full image size and area over bounding box size of the adjacent regions to be merged. We also tried other features such as region's bounding box aspect ratio, perimeter, centroid or major (minor) axis, but do not see noticeable improvement.

We use the graph-based segmentation algorithm [12] to produce four initial over-segmentations per image by adopting two color spaces (Lab and HSV) and two different parameters ($k = 50$ and $k = 100$) respectively. This is the same setting as in [26]. We start our merging processes individually from those 4 over-segmentations and combines their proposal results.

To select proposals for each image given a proposal budget, we randomly rank our proposals using the same strategy as [26]. Each proposal is firstly assigned an initial value that equals to its position in the segmentation hierarchy, then we multiply it by a random number to get the priority value for the proposal. All proposals of an image are finally sorted according to their priority values.

We implement our method in Matlab. The average runtime for our method is around 2.8 seconds per image on an Intel 3.4GHz CPU.

4.2. Experiment settings

Following previous methods [2, 26, 7, 28], we conducts our experiments on the PASCAL VOC2007 dataset [10], which consists of 9963 images from 20 categories. Our classifiers are trained on the VOC's segmentation set. We validate the proposed approach in Sec. 4.3, using the validation set to investigate the impact of and determine several important parameters. We report the results compared to other state-of-the-arts on the test set in Sec. 4.4.

Since current proposal methods are tuned on PASCAL's

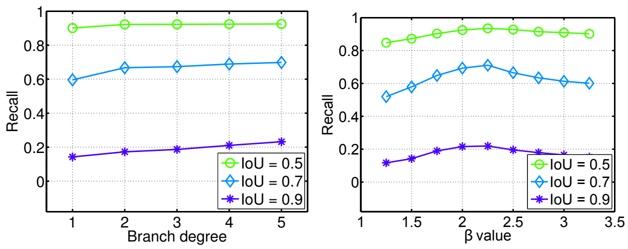

Figure 3. Recall rate on different IoU thresholds while varying the parameter branch degree (left) and β (right) on the third stage.

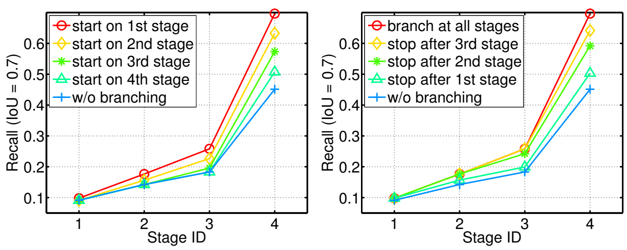

Figure 4. Recall rate of our approach variants at each stage on the IoU threshold of 0.7. Left: branching begins at different stages. Right: branching stops at different stages.

20 object categories, there's some concern about whether these methods can generalize well for novel object categories. Hence, in Sec. 4.4, we also evaluate our method on ImageNet2013 [25] validation set, which consists of over 20000 images with 200 object categories.

4.3. Investigation of parameters

In this section, we investigate the effect of our multi-branch approach. Because it is hard to tell by which branch an object is found if we combine object proposals using multiple initial over-segmentations, in this section we only use one initial over-segmentation (Lab color space and $k = 50$) for clarity.

Impact of important parameters. To evaluate our algorithm's behavior upon the branch degree and classifier complementary factor β, which both influence the branch diversity, we vary their values on each stage respectively and then compute the recall rate obtained on different intersection over union[2] (IoU) thresholds. Note that we just show the result on the third stage in Fig. 3 for brevity, since the results on other stages are similar. As shown in Fig. 3 (left), increasing branch degree can continuously improve the recall rate, but little improvement is observed when branch degree is larger than 2. From Fig. 3 (right), we can see that maximum recall rate is reached when β is around 2. Thus we use a four-stage, two-branch (using 2 complementary classifiers) structure in our experiment below and set β to 2 on each stage.

Evaluation of multi-stage branching. We evaluate our multi-stage branching by comparing it against a baseline without branching, which only uses the first classifier in each stage. Then we produce other variants by starting branching on different stages or stoping branching after one certain stage, respectively. Finally, we compare our approach to all these variants. Detailed recall rates using the IoU threshold of 0.7 on each stage are reported in Fig. 4.

We make two important observations from Fig. 4. Firstly, branching can significantly improve the performance. Specifically, after using branching, the recall rate increases from 0.45 to 0.7, and achieves more than 50% improvement. Secondly, branching at different stages can always make a progress and removing a branch at any stage will lead to performance drop. Therefore, it is necessary to branch at all stages to obtain the best performance.

4.4. Comparison with the state-of-the-art

We compare with recent state-of-the-art methods, including selective search (SS) [26], geodesic object proposals (GOP) [19], global and local search (GLS) [23], edge boxes (EB) [28] and binarized normed gradients (BING) [7]. Notice that EB and BING are scoring-based methods, while the others are segment-based.

In the experiment, we use the setting EdgeBox70 (optimized for IoU=0.7) for EB; $N_S = 200, N_\Lambda = 15$ for GOP, and default settings for the rest of the methods. These settings should yield the best overall performance for them.

Object proposal evaluation. The standard Recall-#Proposal, Recall-IoU and AUC-#Proposal Curves are employed to evaluate the performance of each method. In the Recall-#Proposal Curve, the IoU threshold is fixed first and then recall rates are computed when proposal number increases. Instead, the Recall-IoU Curve shows the performance of top N boxes proposed on the different IoU thresholds. Finally, AUC-#proposal Curve shows the area under curve (AUC) values of Recall-IoU curves at different window proposal number thresholds.

Typically an IoU threshold of 0.5 is used to judge whether an object is successfully detected by the detector in object detection tasks. However as observed in recent works [28, 19, 18] that an object proposal with 0.5 IoU is too loose to fit the ground truth object, which usually leads to the failure of later object detectors. In order to achieve good detection results, an object proposal with higher IoU such as 0.7 or 0.8 is desired. Therefore, the recall performance measured under a tight IoU threshold is more important than under a loss threshold like 0.5. Moreover, it's shown in [17] that the mAP rate of a detector is most related to the AUC score of the object proposal method. This is confirmed in our latter experiment in Sec 4.4.

[2]The Intersection over Union (IoU) [10] is typically used to measure the accuracy of a window proposal. Is is defined as the intersection area of a proposal with its nearest ground truth box divided by their union.

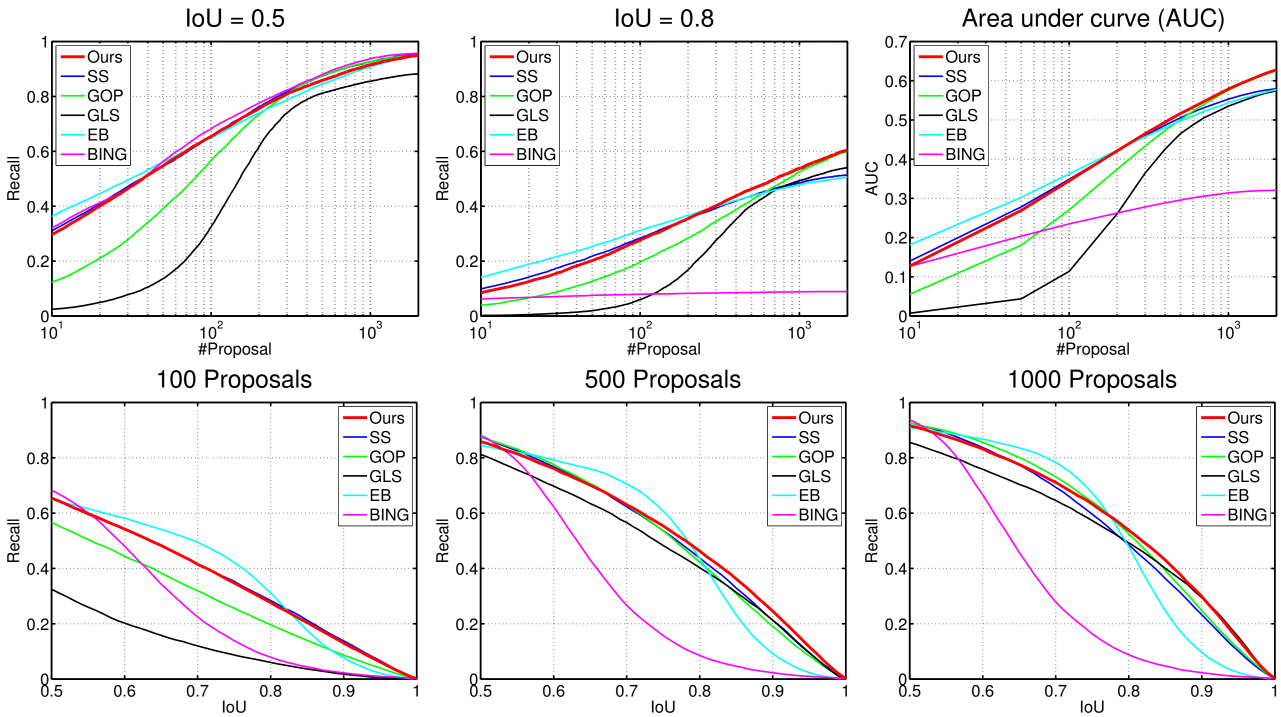

Figure 5. Comparison of our method to various state-of-the-art methods on PASCAL VOC2007 test set. Top left & middle: recall rate no less than a fixed IoU threshold for different proposal numbers. Top right: Area under curve (AUC) for different proposal numbers. Bottom: recall rate using different IoU thresholds for fixed proposal budgets.

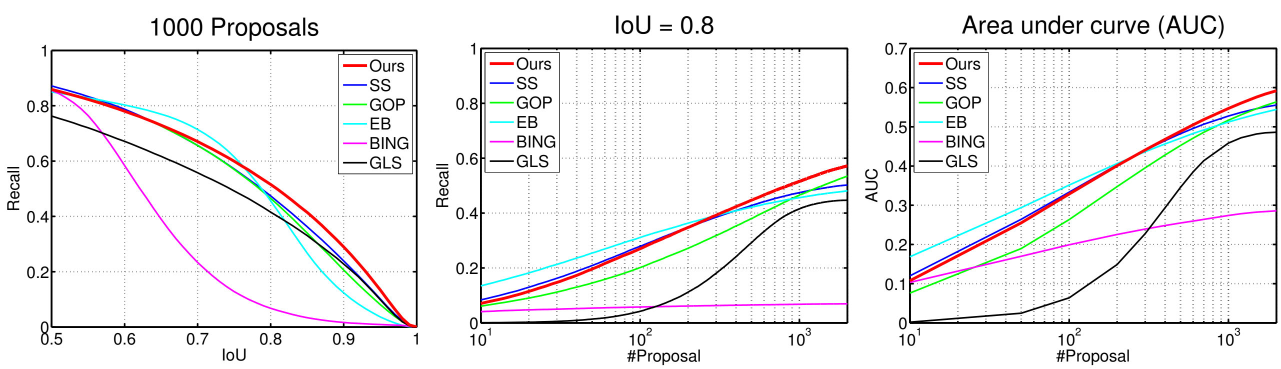

Figure 6. Comparison on ImageNet2013 validation set.

As shown in Fig. 5, scoring-based methods (EB and BING) are competitive on the IoU threshold of 0.5 due to their effective scoring algorithms. However, as the threshold increases, their curves drops significantly. By contrast, all segment-based approaches show favourable performance consistently when IoU threshold is changed. We note that our method, SS and GOP always rank top at different IoU thresholds and top N proposal budgets. Ours outperforms SS and GOP under tight IoU thresholds, and has the best AUC-#Proposal Curve.

As mentioned previously, SS [26] is the most related work with ours. To further understand why our branching can improve, we compare our method with SS on different object sizes. Since SS on average generates 2000 proposals per image, we report our proposal results using the first 2000 proposals for fairness. Fig. 8 shows their proposal number and recall rate. We can see that ours outperforms SS [26] for large objects (bounding box area $> 10^4$), and SS is only moderately better on small objects. This is because our branching increases the search space for large complex objects, while SS concentrates its search more on small objects (see Fig. 8 left).

In addition, in order to show whether ours and other methods generalize well outside PASCAL's 20 object cat-

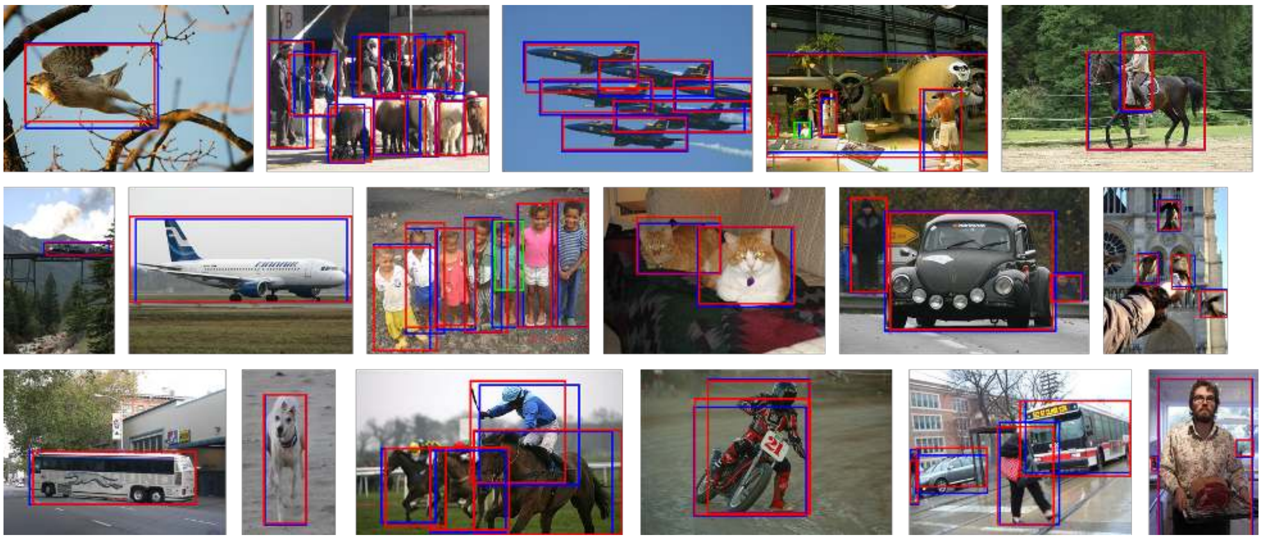

Figure 7. Qualitative examples of our proposal results. Blue: ground truth boxes. Red: our proposal windows with $IoU \geq 0.7$. Green: our proposal windows with $IoU < 0.7$.

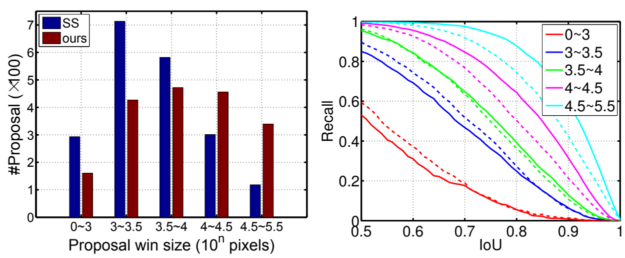

Figure 8. Comparison of our method and SS [26] on different object sizes (10^n pixels). Left: proposal number of two methods. Right: recall rate of ours (solid line) and SS (dashed line).

	mAP (%)	recall AUC	avg #prop.
BING [7]	37.9	0.321	1927
EB [28]	52.0	0.602	4049
GLS [23]	52.6	0.580	**1787**
SS [26]	54.2	0.584	2008
GOP [19]	53.8	**0.648**	2954
Ours(fast)	**54.4**	0.592	**1315**
Ours	**55.2**	**0.648**	2506

Table 1. R-CNN object detection results using various generic object proposal methods on the PASCAL VOC2007 test dataset. Mean average precision (mAP), area under the curve (AUC) of recall rate, average proposal number per image are reported.

egories, we further conduct evaluation on ImageNet2013 validation set. As shown in Fig. 6, our method achieves the most competitive results among all the compared methods. Moreover, it's worth noticing that our approach's improvement over other methods on ImageNet is larger than that on PASCAL VOC.

Using object proposal for object detection. In object detection task, the object proposal algorithm provides candidate windows for subsequent object detectors. Therefore, it is essential to evaluate the performance of proposal methods by the final detection precision. For this purpose, we directly evaluate the state-of-the-art R-CNN [14] detector's performance using different object proposal methods as detector's input. We turn off the bounding box regression process described in [14] in order to compare the original performance of different proposal methods on a fair basis.

In addition to our approach's standard version, we evaluate a fast version of ours, which only combines 2 (Lab, $k = 50, 100$) out of the 4 initial oversegmentations.

Detailed results are reported in Table 1. As seen, our approach's standard version achieves both the best mAP (55.2) and the highest AUC (0.648), while our approach's fast version generates the fewest object proposals (1315) and still obtains better mAP (54.4) than all the other state-of-the-art object proposal methods. Although GOP has the same AUC as ours, its proposal numbers is about 500 more.

5. Conclusions

We propose a novel object proposal algorithm that learns multiple complementary merging strategies to branch the search for object proposals. The approach achieves the state-of-the-art performance under most object proposal quality measurements. Future work includes exploiting more effective features and applying the multi-branch idea to semantic image segmentation.

Acknowledgements The work of the first and fourth author was supported by the National Basic Research Program of China (Grant No.2015CB856004) and the National Natural Science Foundation of China (Grant No.61272251).

The work of the second, third and fifth author was supported by The National Science Foundation of China (No.61272276, No.61305091), the National Twelfth Five-year Plan Major Science and Technology Project of China (No.2012BAC11B01-04-03), Special Research Fund of Higher Colleges Doctorate (No.20130072110035), the Fundamental Research Funds for the Central Universities (No.2100219038), and Shanghai Pujiang Program (No.13PJ1408200).

References

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

[2] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(11):2189–2202, 2012. 2, 5

[3] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik. From contours to regions: An empirical evaluation. In *CVPR*, 2009. 1

[4] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 33(5):898–916, 2011. 1, 3

[5] P. Arbeláez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *CVPR*, 2014. 1, 3

[6] J. Carreira and C. Sminchisescu. CPMC: Automatic Object Segmentation Using Constrained Parametric Min-Cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(7):1312–1328, 2012. 1, 3

[7] M.-M. Cheng, Z. Zhang, W.-Y. Lin, and P. H. S. Torr. BING: Binarized Normed Gradients for Objectness Estimation at 300fps. In *CVPR*, 2014. 1, 3, 5, 6, 8

[8] P. Dollár and C. L. Zitnick. Structured forests for fast edge detection. In *ICCV*, 2013. 3

[9] I. Endres and D. Hoiem. Category-independent object proposals with diverse ranking. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 36(2):222–234, 2014. 1, 3

[10] M. Everingham, L. J. V. Gool, C. K. I. Williams, J. M. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision (IJCV)*, 88(2):303–338, 2010. 2, 4, 5, 6

[11] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008. 4

[12] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision (IJCV)*, 59(2):167–181, 2004. 1, 3, 5

[13] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997. 2, 4

[14] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 1, 2, 3, 8

[15] S. Gupta, R. B. Girshick, P. A. Arbeláez, and J. Malik. Learning rich features from RGB-D images for object detection and segmentation. In *ECCV*, 2014. 1

[16] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014. 1, 3

[17] J. Hosang, R. Benenson, P. Dollár, and B. Schiele. What makes for effective detection proposals? *arXiv:1502.05082*, 2015. 2, 6

[18] J. Hosang, R. Benenson, and B. Schiele. How good are detection proposals, really? In *BMVC*, 2014. 2, 6

[19] P. Krähenbühl and V. Koltun. Geodesic object proposals. In *ECCV*, 2014. 1, 3, 6, 8

[20] W. G. Kropatsch. Building irregular pyramids by dual-graph contraction. *IEE Proceedings - Vision, Image and Signal Processing*, 142(6):366–374, 1995. 1

[21] S. Manen, M. Guillaumin, and L. V. Gool. Prime object proposals with randomized prims algorithm. In *ICCV*, 2013. 1, 3

[22] E. Rahtu, J. Kannala, and M. B. Blaschko. Learning a category independent object detection cascade. In *ICCV*, 2011. 1, 3

[23] P. Rantalankila, J. Kannala, and E. Rahtu. Generating object segmentation proposals using global and local search. In *CVPR*, 2014. 1, 2, 3, 6, 8

[24] Z. Ren and G. Shakhnarovich. Image segmentation by cascaded region agglomeration. In *CVPR*, 2013. 1, 2

[25] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. 2014. 6

[26] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision (IJCV)*, 104(2):154–171, 2013. 1, 2, 3, 5, 6, 7, 8

[27] D. J. Weiss and B. Taskar. Scalpel: Segmentation cascades with localized priors and efficient learning. In *CVPR*, 2013. 1

[28] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, 2014. 1, 3, 5, 6, 8