Region Proposal by Guided Anchoring

Jiaqi Wang[1*] Kai Chen[1*] Shuo Yang[2] Chen Change Loy[3] Dahua Lin[1]
[1]CUHK - SenseTime Joint Lab, The Chinese University of Hong Kong
[2]Amazon Rekognition [3]Nanyang Technological University
{ck015,wj017,dhlin}@ie.cuhk.edu.hk shuoy@amazon.com ccloy@ntu.edu.sg

arXiv:1901.03278v1 [cs.CV] 10 Jan 2019

Abstract

Region anchors are the cornerstone of modern object detection techniques. State-of-the-art detectors mostly rely on a dense anchoring scheme, where anchors are sampled uniformly over the spatial domain with a predefined set of scales and aspect ratios. In this paper, we revisit this foundational stage. Our study shows that it can be done much more effectively and efficiently. Specifically, we present an alternative scheme, named Guided Anchoring, which leverages semantic features to guide the anchoring. The proposed method jointly predicts the locations where the center of objects of interest are likely to exist as well as the scales and aspect ratios at different locations. On top of predicted anchor shapes, we mitigate the feature inconsistency with a feature adaption module. We also study the use of high-quality proposals to improve detection performance. The anchoring scheme can be seamlessly integrated to proposal methods and detectors. With Guided Anchoring, we achieve 9.1% *higher recall on MS COCO with* 90% *fewer anchors than the RPN baseline. We also adopt Guided Anchoring in Fast R-CNN, Faster R-CNN and RetinaNet, respectively improving the detection mAP by* 2.2%, 2.7% *and* 1.2%. [1]

1. Introduction

Anchors are regression references and classification candidates to predict proposals (for two-stage detectors) or final bounding boxes (for single-stage detectors). Modern object detection pipelines usually begin with a large set of densely distributed anchors. Take Faster RCNN [30], a popular object detection framework, for instance, it first generates region proposals from a dense set of anchors and then classifies them into specific classes and refines their locations via bounding box regression.

There are two general rules for a reasonable anchor design: *alignment* and *consistency*. Firstly, to use convolutional features as anchor representations, anchor centers need to be well aligned with feature map pixels. Secondly, the receptive field and semantic scope are consistent in different regions of a feature map, so the scale and shape of anchors across different locations should be consistent. Sliding window is a simple and widely adopted anchoring scheme following the rules. For most detection methods, the anchors are defined by such a *uniform* scheme, where every location in a feature map is associated with k anchors with predefined scales and aspect ratios.

Anchor-based detection pipelines have been shown effective in both benchmarks [7, 22, 8, 5] and real-world systems. However, the uniform anchoring scheme described above is not necessarily the optimal way to prepare the anchors. This scheme can lead to two difficulties: (1) A neat set of anchors of fixed aspect ratios has to be predefined for different problems. A wrong design may hamper the speed and accuracy of the detector. (2) To maintain a sufficiently high recall for proposals, a large number of anchors are needed, while most of them correspond to *false* candidates that are irrelevant to the object of interests. Meanwhile, large number of anchors can lead to significant computational cost especially when the pipeline involves a heavy classifier in the proposal stage.

In this work, we present a more effective method to prepare anchors, with the aim to mitigate the issues of hand-picked priors. Our method is motivated by the observation that objects are not distributed evenly over the image plane. The scale of an object is also closely related to the imagery content, its location and geometry of the scene. Following this intuition, our method generates sparse anchors in two steps: first identifying sub-regions that may contain objects and then determining the scales and aspect ratios at different locations.

Learnable anchor shapes are promising, but it breaks the aforementioned rule of consistency, thus presents a new challenge for learning anchor representations for accurate classification and regression, Scales and aspect ratios of anchors are now variable instead of fixed, so different feature map pixels have to learn adaptive representations that fit to

*Equal contribution.
[1]Code will be available.

the corresponding anchors. To solve this problem, we introduce an effective module to adapt the features based on anchor geometry.

We formulate a Guided Anchoring Region Proposal Network (GA-RPN) with the aforementioned guided anchoring and feature adaptation scheme. Thanks to the dynamically predicted anchors, our approach achieves 9.1% higher recall with 90% substantially fewer anchors than the RPN baseline that adopts dense anchoring scheme. By predicting the scales and aspect ratios instead of fixing them based on a predefined list, our scheme handles tall or wide objects more effectively. Besides region proposals, the guided anchoring scheme can be easily integrated into any detectors that depend on anchors. Consistent performance gains can be achieved with our scheme. For instance, GA-Fast-RCNN, GA-Faster-RCNN and GA-RetinaNet improve overall mAP by 2.2%, 2.7% and 1.2% respectively on COCO dataset over their baselines with sliding window anchoring. Furthermore, we explore the use of high-quality proposals, and propose a fine-tuning schedule using GA-RPN proposals, which can improve the performance of any trained models, *e.g.*, it improves a fully converged Faster R-CNN model from 37.4% to 39.6%, in only 3 epochs.

The main contributions of this work lie in several aspects. (1) We propose a new anchoring scheme with the ability to predict non-uniform and arbitrary shaped anchors other than dense and predefined ones. (2) We formulate the joint anchor distribution with two factorized conditional distributions, and design two modules to model them respectively. (3) We study the importance of aligning features with the corresponding anchors, and design a feature adaption module to refine features based on the underlying anchor shapes. (4) We investigate the use of high quality proposals for two-stage detectors, and propose a scheme to improve the performance of trained models.

2. Related Work

Classical object detectors. The sliding window paradigm, in which a classifier is applied on a dense image grid, has a long history and dominates the field of object detection in classic computer vision. Many early progress is closely related to more powerful handcrafted features, such as Histogram of Oriented Gradients (HOG) [4], SIFT [24] and Integral Channel Features [6].

Two-stage object detectors. The two-stage approach has been the leading paradigm in the modern era of object detection. The first stage generates a sparse set of object proposals, and the second stage classifies the proposals as well as refine the coordinates. In early explorations [13, 12], object proposals are generated by external modules [38, 33]. Faster R-CNN [30] introduces the Region Proposal Network (RPN) as object proposal component. It uses a small fully convolutional network to map each sliding window anchor to a low-dimensional feature. This design is widely adopted in later two-stage methods [2, 20, 14].

Single-stage object detectors. Compared to two-stage approaches, the single-stage pipeline skips object proposal generation and predicts bounding boxes and class scores in one evaluation. Although the proposal step is omitted, single-stage methods still use anchor boxes produced by sliding window. For instance, SSD [23] and DenseBox [16] generate anchors densely from feature maps and evaluate them like a multi-class RPN. RetinaNet [21] shares many similarities with SSD, added with focal loss and Feature Pyramid Network (FPN) [20] to address foreground-background class imbalance and small objects. YOLOv2[29] adopt sliding window anchors to for classification and spatial location prediction so as to achieve a higher recall than its precedent.

Cascaded classification and regression. There have been attempts [9, 10, 26, 35, 36, 37, 1] that apply cascade architecture to reject easy samples at early layers or stages, and regress bounding boxes iteratively for progressive refinement.

Comparison and difference. We summarize the differences between the proposed method and conventional methods as follows. (i) Primarily, previous methods (single-stage, two-stage and multi-stage) still rely on dense and uniform anchors by sliding window. We discard the sliding window scheme and propose a better counterpart to guide the anchoring and generate sparse anchors, which has not been explored before. (ii) Cascade detectors adopt more than one stage to refine detection bounding boxes progressively, which usually leads to more model parameters and a decrease in inference speed. These methods adopts RoI Pooling or RoI Align to extract aligned features for bounding boxes, which is too expensive for proposal generation or single-stage detectors. (iii) Anchor-free methods [16, 17, 28] usually have simple pipelines and produce final detection results within a single stage. Due to the absence of anchors and further anchor-based refinement, they lack the ability to deal with complex scenes and cases. Our focus is the sparse and non-uniform anchoring scheme and use of high-quality proposals to boost the detection performance. Towards this goal, we have to solve the misalignment and inconsistency issues which are specific to anchor-based methods. Moreover, [17] assumes segmentation mask annotations as supervision its size prediction is used to weighted sum the output of multiple scale-specific networks. (iv) Some single-shot detectors [36, 34] refine anchors by multiple regression and classification. Our method differs from them significantly. We do not refine anchors progressively, instead, we predict the distribution of anchors, which is factorized as locations and shapes. Conventional methods fail to consider the alignment between anchors and features so they regress anchors (represented

by $[x, y, w, h]$) for multiple times and breaks the alignment as well as consistency. On the contrary, we emphasize the importance of the two rules, so we only predict anchor shapes but fix anchor centers and adapt features based on the predicted shapes. Their classification target is to identify whether an anchor has a larger overlap with some ground truth object than a threshold. But our location branch is to predict whether a point is close to object centers instead of labels of any specific anchors.

3. Guided Anchoring

Anchors are the basis in modern object detection pipelines. Mainstream frameworks, including two-stage and single-stage methods, mostly rely on a *uniform* arrangement of anchors. Specifically, a set of anchors with predefined scales and aspect ratios will be deployed over a feature map of size $W \times H$, with a stride of s. This scheme is inefficient, as many of the anchors are placed in regions where the objects of interest are unlikely to exist. In addition, such hand-picked priors unrealistically assume a set of fixed shape (*i.e.*, scale and aspect ratio) for objects.

In this work, we aim to develop a more efficient anchoring scheme to arrange the anchors with learnable shapes, considering the non-uniform distribution of objects' locations and shapes. The guided anchoring scheme works as follows. The location and the shape of an object can be characterized by a 4-tuple in the form of (x, y, w, h), where (x, y) is the spatial coordinate of the center, w the width, and h the height. Suppose we draw an object from a given image I, then its location and shape can be considered to follow a distribution conditioned on I, as follows:

$$p(x, y, w, h|I) = p(x, y|I)p(w, h|x, y, I). \quad (1)$$

This factorization captures two important intuitions: (1) given an image, objects may only exist in certain regions; and (2) the shape, *i.e.*, scale and aspect ratio, of an object closely relates to its location.

Following this formulation, we devise an anchor generation module as shown in the red dashed box of Figure 1. This module is a network comprised of two branches for location and shape prediction, respectively. Given an image I, we first derive a feature map F_I. On top of F_I, the *location prediction* branch yields a probability map that indicates the possible locations of the objects, while the *shape prediction* branch predicts location-dependent shapes. Given the outputs from both branches, we generate a set of *anchors* by choosing the locations whose predicted probabilities are above a certain threshold and the most probable shape at each of the chosen locations. As the anchor shapes can vary, the features at different locations should capture the visual content within different ranges. With this taken into consideration, we further introduce a *feature adaptation* module, which adapts the feature according to the anchor shape.

The anchor generation process described above is based on a single feature map. Recent advances in object detection [20, 21] show that it is often helpful to operate on multiple feature maps at different levels. Hence, we also develop a multi-level anchor generation scheme, which collect anchors at multiple feature maps, following the FPN architecture [20]. Note that in our design, the anchor generation parameters are shared across all involved feature levels thus the scheme is parameter-efficient.

3.1. Anchor Location Prediction

As shown in Figure 1, the *anchor location prediction* branch yields a probability map $p(\cdot|F_I)$ of the same size as the input feature map F_I, where each entry $p(i, j|F_I)$ corresponds to the location with coordinate $((i + \frac{1}{2})s, (j + \frac{1}{2})s)$ on I, where s is the stride of feature map, *i.e.*, the distance between neighboring anchors. The entry's value indicates the probability of an object's center existing at that location.

In our formulation, the probability map $p(i, j|F_I)$ is predicted using a sub-network $\mathcal{N}_L$. This network applies a 1×1 convolution to the base feature map F_I to obtain a map of objectness scores, which are then converted to probability values via an element-wise sigmoid function. While a deeper sub-network can make more accurate predictions, we found empirically that a convolution layer followed by a sigmoid transform strikes a good balance between efficiency and accuracy.

Based on the resultant probability map, we then determine the active regions where objects may possibly exist by selecting those locations whose corresponding probability values are above a predefined threshold ϵ_L. This process can filter out 90% of the regions while still maintaining the same recall. As illustrated in Figure 4, regions like sky and ocean are excluded, while anchors concentrate densely around persons and surfboards. Since there is no need to consider those excluded regions, we replace the ensuing convolutional layers by *masked convolution* [19, 31] for more efficient inference.

3.2. Anchor Shape Prediction

After identifying the possible locations for objects, our next step is to determine the shape of the object that may exist at each location. This is accomplished by the *anchor shape prediction* branch (Fig. 1(c)). This branch is very different from conventional bounding box regression, since it does not change the anchor positions and will not cause misalignment between anchors and anchor features. Concretely, given a feature map F_I, this branch will predict the best shape (w, h) for each location, *i.e.*, the shape that may lead to the highest IoU with the nearest groundtruth bounding box.

While our goal is to predict the values of the width w and the height h, we found empirically that directly predicting

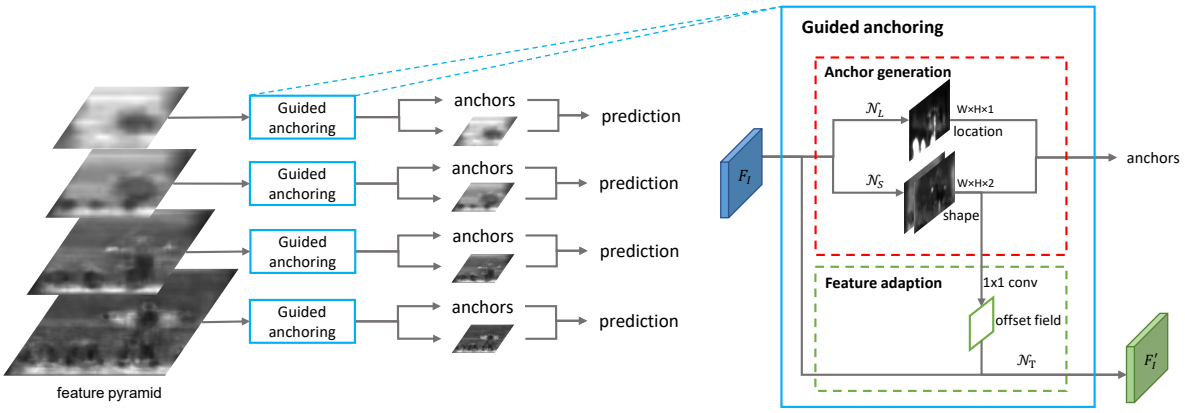


Figure 1: An illustration of our framework. For each output feature map in the feature pyramid, we use an anchor generation module with two branches to predict the anchor location and shape, respectively. Then a feature adaption module is applied to the original feature map to make the new feature map aware of anchor shapes.

these two numbers is not stable, due to their large range. Instead, we adopt the following transformation:

$$w = \sigma \cdot s \cdot e^{dw}, \quad h = \sigma \cdot s \cdot e^{dh}. \tag{2}$$

The shape prediction branch will output dw and dh , which will then be mapped to (w, h) as above, where s is the stride and σ is an empirical scale factor ($\sigma = 8$ in our experiments). This nonlinear transformation projects the output space from approximate $[0, 1000]$ to $[-1, 1]$, leading to an easier and stable learning target. In our design, we use a sub-network $\mathcal{N}_S$ for shape prediction, which comprises a 1×1 convolutional layer that yields a two-channel map that contains the values of dw and dh, and an element-wise transform layer that implements Eq.(2).

Note that this design differs essentially from the conventional anchoring schemes in that every location is associated with just one anchor of the dynamically predicted shape instead of a set of anchors of predefined shapes. Our experiments show that due to the close relations between locations and shapes, our scheme can achieve higher recall than the baseline scheme. It is also worth noting that our scheme allows *arbitrary* aspect ratios, and thus it is capable of capturing those extremely tall or wide objects better, such as trains and snowboards.

3.3. Anchor-Guided Feature Adaptation

In the conventional RPN or single stage detectors where the sliding window scheme is adopted, anchors are uniform on the whole feature map, *i.e.*, they share the same shape and scale in each position. Thus the feature map can learn consistent representation. In our scheme, however, the shape of anchors varies across locations. Under this condition, we find that it may not be a good choice to follow the previous convention [30], in which a fully convolutional classifier is applied uniformly over the feature map. Ideally, the feature for a large anchor should encode the content over a large region, while those for small anchors should have smaller scopes accordingly. Following this intuition, we further devise an *anchor-guided feature adaptation* component, which will transform the features at individual location based on the underlying anchor shapes, as

$$\mathbf{f}_i' = \mathcal{N}_T(\mathbf{f}_i, w_i, h_i), \tag{3}$$

where $\mathbf{f}_i$ is the feature at the i-th location, (w_i, h_i) is the corresponding anchor shape. For such a location-dependent transformation, we adopt a 3×3 deformable convolutional layer [3] to implement $\mathcal{N}_T$. As shown in Figure 1, we first predict an offset field from the output of anchor shape prediction branch, and then apply deformable convolution to the original feature map with the offsets to obtain f_I'. On top of the adapted features, we can then perform further classification and bounding-box regression.

3.4. Training

Joint objective. The proposed framework is optimized in an end-to-end fashion using a multi-task loss. Apart from the conventional classification loss $\mathcal{L}_{cls}$ and regression loss $\mathcal{L}_{reg}$, we introduce two additional losses for the anchor localization $\mathcal{L}_{loc}$ and anchor shape prediction $\mathcal{L}_{shape}$, which are covered in previous sections. They are jointly optimized with the following loss.

$$\mathcal{L} = \lambda_1 \mathcal{L}_{loc} + \lambda_2 \mathcal{L}_{shape} + \mathcal{L}_{cls} + \mathcal{L}_{reg}. \tag{4}$$

Anchor location targets. To train the anchor localization branch, for each image we need a binary label map where 1 represents a valid location to place an anchor and 0 otherwise. In this work, we employ ground-truth bounding boxes for guiding the binary label map generation. In particular, we wish to place more anchors around the vicinity of an object's center, while fewer of them far from

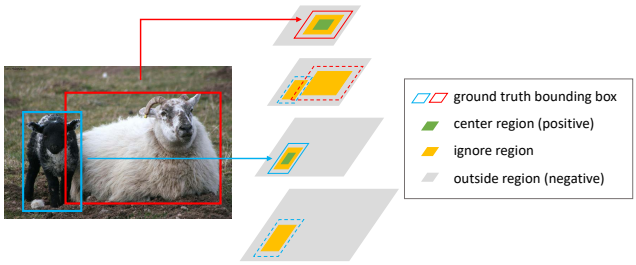


Figure 2: Anchor location target for multi-level features. We assign ground truth objects to different feature levels according to their scale, and define CR, IR and OR respectively. (Best viewed in color.)

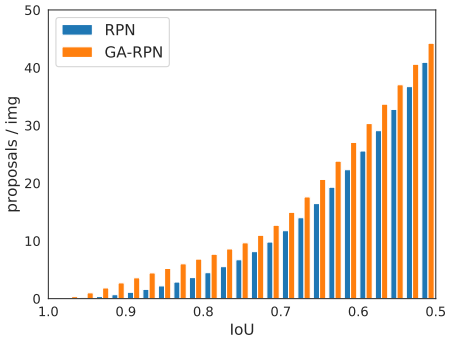


Figure 3: IoU distribution of RPN and GA-RPN proposals. We show the accumulated proposal number with increasing IoUs.

the center. Firstly, we map the ground-truth bounding box (x_g, y_g, w_g, h_g) to the corresponding feature map scale, and obtain (x'_g, y'_g, w'_g, h'_g). We denote $\mathcal{R}(x, y, w, h)$ as the rectangular region whose center is (x, y) and the size of $w \times h$. Anchors are expected to be placed close to the center of ground truth objects to obtain larger initial IoU, thus we define three types of regions for each box.

(1) The center region $CR = \mathcal{R}(x'_g, y'_g, \sigma_1 w', \sigma_1 h')$ defines the center area of the box. Pixels in CR are assigned as positive samples.

(2) The ignore region $IR = \mathcal{R}(x'_g, y'_g, \sigma_2 w', \sigma_2 h') \backslash CR$ is a larger $(\sigma_2 > \sigma_1)$ region excluding CR. Pixels in IR are marked as "ignore" and exluded during training.

(3) The outside region OR is the whole feature map excluding CR and IR. Pixels in OR are regarded as negative samples.

Previous work [16] proposed the "gray zone" for balanced sampling, which has a similar definition to our location targets but only works on a single feature map. Since we uses multiple feature levels from FPN, we also consider the influence of adjacent feature maps. Specifically, each level of feature map should only target objects of a specific scale range, so we assign CR on a feature map only if the feature map matches the scale range of the targeted object. The same regions of adjacent levels are set as IR, as shown in Figure 2. CR usually accounts for a small portion of the whole feature map, so we use Focal Loss [21] to train the location branch.

Anchor shape targets. There are two steps to determine the best shape target for each anchor. First, we need to match the anchor to a ground-truth bounding box. Next, we will compute the optimal shape $\hat{w}$ and $\hat{h}$ which can maximize the IoU between the anchor and the matched ground-truth bounding box.

Previous works [30] assign a candidate anchor to the ground truth bounding box that yields the largest IoU value with the anchor. However, this process is not applicable in our case, since w and h of our anchors are not predefined but variables. To overcome this problem, we define the IoU between a variable anchor $a_{wh} = (x_0, y_0, w, h)$ and a ground truth bounding box $\text{gt} = (x_g, y_g, w_g, h_g)$ as follows, denoted as vIoU.

$$\text{vIoU}(a_{wh}, \text{gt}) = \max_{w>0, h>0} \text{IoU}_{normal}(a_{wh}, \text{gt}) \quad (5)$$

where IoU_{normal} is the typical definition of IoU and w and h are variables. Note that for an arbitrary anchor location (x_0, y_0) and ground-truth gt, the analytic expression of $\text{vIoU}(a_{wh}, \text{gt})$ is complicated, and hard to be implemented efficiently in an end-to-end network. Therefore we use an alternative way to approximate it. Given (x_0, y_0), we sample some common values of w and h to simulate the enumeration of all w and h. Then we calculate the IoU of these sampled anchors with gt, and use the maximum as an approximation of $\text{vIoU}(a_{wh}, \text{gt})$. In our experiments, we sample 9 pairs of (w, h) of different scales and aspect ratios as a tradeoff for accuracy and efficiency. We finally assign the location (x_0, y_0) to the ground-truth that yields the largest vIoU.

Instead of directly regressing the optimal shape $\hat{w}$ and $\hat{h}$, we adopt a variant of bounded iou loss [32] to maximize the IoU between the anchor and corresponding ground truth bounding box. The loss is defined almost the same as the original paper except that we only optimize w and h instead of $\{x, y, w, h\}$, since the anchor location is fixed.

3.5. The Use of High-quality Proposals

RPN enhanced by the proposed guided anchoring scheme (GA-RPN) can generate much higher quality proposals than the conventional RPN. We explore how to boost the performance of conventional two-stage detectors, through the use of such high quality proposals. Firstly, we study the IoU distribution of proposals generated by RPN and GA-RPN, as shown in Figure 3. There are two significant advantages of GA-RPN proposals over RPN proposals: (1) the number of positive proposals are larger, and (2) the ratio of high-IoU proposals are more significant. A straightforward idea is to replace RPN in existing models with the

proposed GA-RPN and train the model end-to-end. However, this problem is non-trivial and adopting exactly the same settings as before can only bring limited gain (*e.g.*, less than 1 point). From our observation, the pre-requisite of using high-quality proposals is to adapt the distribution of training samples in accordance to the proposal distribution. Consequently, we set a higher positive/negative threshold and use fewer samples when training detectors with GA-RPN compared to RPN.

Besides end-to-end training, we find that GA-RPN proposals is capable of boosting a trained two-stage detector by a fine-tuning schedule. Specifically, given a trained model, we discard the proposal generation component, *e.g.*, RPN, and use pre-computed GA-RPN proposals to finetune it for several epochs (3 epochs by default). GA-RPN proposals are also used for inference. This simple fine-tuning scheme can further improve the performance by a large margin, with only a time cost of a few epochs.

4. Experiments

4.1. Experimental Setting

Dataset. We perform experiments on the challenging MS COCO 2017 benchmark [22]. We use the *train* split for training and report the performance on *val* split. Detection results are reported on *test-dev* split.

Implementation details. We use ResNet-50 [15] with FPN [20] as the backbone network for all experiments, if not otherwise specified. As a common convention, we resize images to the scale of 1333×800, without changing the aspect ratio. We set $\sigma_1 = 0.2, \sigma_2 = 0.5$ when defining CR and IR for anchor location prediction. In the multi-task loss function, we simply use $\lambda_1 = 1, \lambda_2 = 0.1$ to balance the location and shape prediction branches. We use synchronized SGD over 8 GPUs with a total of 16 images per minibatch (2 images per GPU). We train 12 epochs in total with an initial learning rate of 0.02, and decrease the learning rate by 0.1 at epoch 8 and 11. The runtime is measured on TITAN X GPUs.

Evaluation metrics. The results of RPN are measured with Average Recall (AR), which is the average of recalls at different IoU thresholds (from 0.5 to 0.95). AR for 100, 300, and 1000 proposals per image are denoted as AR_{100}, AR_{300} and AR_{1000}. The AR for small, medium, and large objects (AR_S, AR_M, AR_L) are computed for 100 proposals. Detection results are evaluated with the standard COCO metric, which averages mAP of IoUs from 0.5 to 0.95.

4.2. Results

We first evaluate our anchoring scheme by comparing the recall of GA-RPN with the RPN baseline and previous state-of-the-art region proposal methods. Meanwhile, we compare with some variants of RPN. "RPN+9 anchors" denotes using 3 scales and 3 aspect ratios in each feature level, while baselines use only 1 scale and 3 aspect ratios, following [20]. "RPN+Iterative" denotes applying two RPN heads consecutively, with an additional 3×3 convolutional layers between them. "RefineRPN" denotes a similar structure to [36], where anchors are regressed and classified twice with features before and after FPN.

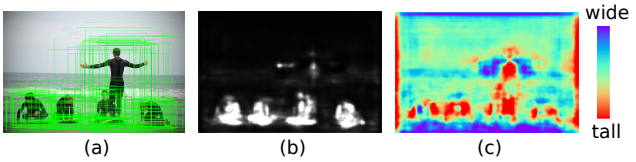


Figure 4: Anchor prediction results. (a) input image and predict anchors; (b) predicted anchor location probability map; (c) predicted anchor aspect ratio.

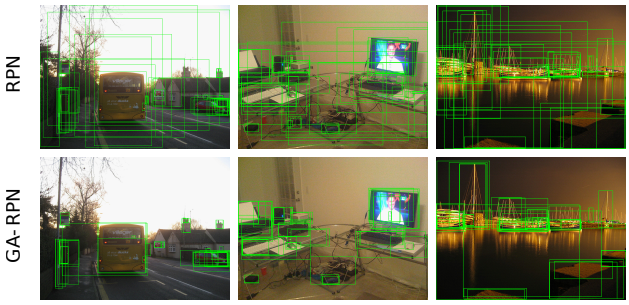


Figure 5: Examples of RPN proposals (top row) and GA-RPN proposals (bottom row).

As shown in Table 1, our method outperforms the RPN baseline by a large margin. Specifically, it improves AR_{300} by 10.5% and AR_{1000} by 9.1% respectively. Notably, GA-RPN with a small backbone can achieve a much higher recall than RPN with larger backbones. Our encouraging results are supported by the qualitative results shown in Figure 4, where we show the sparse and arbitrary shaped anchors and visualize the outputs of two branches. It is observed that the anchors concentrate more on objects and provides a good basis for the ensuing object proposal. In Figure 5, we show some examples of proposals generated upon sliding window anchoring and guided anchoring.

Iterative regression and classification ("RPN+Iterative" and "RefineRPN") only brings limited gain to RPN, which proves the importance of the aforementioned rule of alignment and consistency, and simply refining anchors multiple times is not effective enough. Keeping the center of anchors fixed and adapt features based on anchor shapes are important.

To investigate the generalization ability of guided anchoring and its power to boost the detection performance, we integrate it with both two-stage and single-stage detection pipelines, including Fast R-CNN [12], Faster R-

Table 1: Region proposal results on MS COCO.

Method	Backbone	AR_{100}	AR_{300}	AR_{1000}	AR_S	AR_M	AR_L	runtime (s/img)
SharpMask [27]	ResNet-50	36.4	-	48.2	6.0	51.0	66.5	0.76 (unfair)
GCN-NS [25]	VGG-16 (SyncBN)	31.6	-	60.7	-	-	-	0.10
AttractioNet [11]	VGG-16	53.3	-	66.2	31.5	62.2	77.7	4.00
ZIP [18]	BN-inception	53.9	-	67.0	31.9	63.0	78.5	1.13
RPN	ResNet-50-FPN	47.5	54.7	59.4	31.7	55.1	64.6	**0.09**
	ResNet-152-FPN	51.9	58.0	62.0	36.3	59.8	68.1	0.16
	ResNeXt-101-FPN	52.8	58.7	62.6	37.3	60.8	68.6	0.26
RPN+9 anchors	ResNet-50-FPN	46.8	54.6	60.3	29.5	54.9	65.6	0.09
RPN+Iterative	ResNet-50-FPN	49.7	56.0	60.0	34.7	58.2	64.0	0.10
RefineRPN	ResNet-50-FPN	50.2	56.3	60.6	33.5	59.1	66.9	0.11
GA-RPN	ResNet-50-FPN	**59.2**	**65.2**	**68.5**	**40.9**	**67.8**	**79.0**	0.13

Table 2: Detection results on MS COCO 2017 *test-dev*.

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Fast R-CNN	37.1	59.6	39.7	20.7	39.5	47.1
GA-Fast-RCNN	**39.4**	59.4	42.8	21.6	41.9	50.4
Faster R-CNN	37.1	59.1	40.1	21.3	39.8	46.5
GA-Faster-RCNN	**39.8**	59.2	43.5	21.8	42.6	50.7
RetinaNet	35.9	55.4	38.8	19.4	38.9	46.5
GA-RetinaNet	**37.1**	56.9	40.0	20.1	40.1	48.0

Table 3: Fine-tuning results on a trained Faster R-CNN.

proposals	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
-	37.4	58.9	40.3	20.8	41.1	49.5
RPN	37.3	58.6	40.1	20.4	40.6	49.8
GA-RPN	**39.6**	**59.3**	**43.0**	**22.0**	**42.8**	**52.6**

CNN [30] and RetinaNet [21]. For two-stage detectors, we replace the original RPN with GA-RPN, and for single-stage detectors, the sliding window anchoring scheme is replaced with the proposed guided anchoring. Results in Table 2 show that guided anchoring not only increases the proposal recall of RPN, but also improves the detection performance by a large margin. With guided anchoring, the mAP of these detectors improves by 2.3%, 2.7% and 1.2% respectively.

To further study the effectiveness of high-quality proposals and investigate the fine-tuning scheme, we take a fully converged Faster R-CNN model and finetune it with pre-computed RPN or GA-RPN proposals. We finetune the detector for 3 epochs, with the learning rate of 0.02, 0.002 and 0.0002 respectively. The results are in Table 3 illustrate that RPN proposals cannot bring any gain, while the high-quality GA-RPN proposals bring 2.2% mAP improvement to the trained model with only a time cost of 3 epochs.

Table 4: The effects of each module in our design. L., S., and F.A. denote location, shape, and feature adaptation, respectively.

L.	S.	F.A.	AR_{100}	AR_{300}	AR_{1000}	AR_S	AR_M	AR_L
			47.5	54.7	59.4	31.7	55.1	64.6
✓			48.0	54.8	59.5	32.3	55.6	64.8
	✓		53.8	59.9	63.6	36.4	62.9	71.7
✓	✓		54.0	60.1	63.8	36.7	63.1	71.5
✓	✓	✓	**59.2**	**65.2**	**68.5**	**40.9**	**67.8**	**79.0**

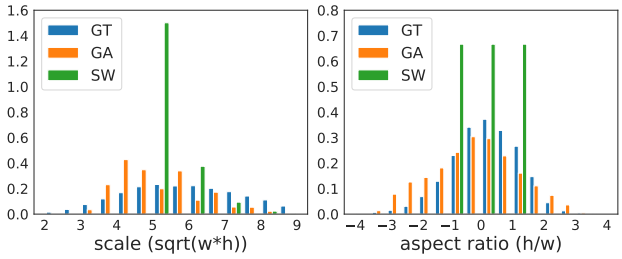
Table 5: Results of different location threshold ϵ_L.

ϵ_L	#anchors/image	AR_{100}	AR_{300}	AR_{1000}	fps
0	75583 (100.0%)	59.2	65.2	68.5	7.8
0.01	22274 (29.4%)	59.2	65.2	68.5	8.0
0.05	5251 (6.5%)	59.1	65.1	68.2	8.2
0.1	2375 (3.2%)	59.0	64.7	67.2	8.2

4.3. Ablation Study

Model design. We omit different components in our design to investigate the effectiveness of each component, including location prediction, shape prediction and feature adaption. Results are shown in Table 4. The shape prediction branch is shown effective which leads to a gain of 4.2%. The location prediction branch brings marginal improvement. Nevertheless, the importance of this branch is reflected in its usefulness of obtaining sparse anchors leading to more efficient inference. The obvious gain brought by the feature adaption module suggests the necessity of rearranging the feature map according to predicted anchor shapes. This module helps to capture information corresponding to anchor scopes, especially for large objects.

Anchor location. The location threshold ϵ_L controls the



(a) (b)

Figure 6: (a) Anchor scale and (b) aspect ratio distributions of different anchoring schemes. The x-axis is reduced to log-space by apply $\log_2(\cdot)$ operator. GT, GA, SW indicates ground truth, guided anchoring, sliding window, respectively.

sparsity of anchor distribution. Adopting different thresholds will yield different number of anchors. To reveal the influence of ϵ_L on efficiency and performance, we vary the threshold and compare the following results: the average number of anchors per image, recall of final proposals and the inference runtime. From Table 5 we can observe that the objectness scores of most background regions are close to 0, so a small ϵ_L can greatly reduce the number of anchors by more than 90%, with only a minor decrease on recall rate. It is noteworthy that the head in RPN is just one convolutional layer, so the speedup is not apparent. Nevertheless, a significant reduction in the number of anchors offers a possibility to perform more efficient inference with a heavier head.

Anchor shape. We compare the set of generated anchors of our method with sliding window anchors of pre-defined shapes. Since our method predicts only one anchor at each location of the feature map instead of k ($k = 3$ in our baseline) anchors of different scales and aspect ratios, the total anchor number is reduced by $\frac{1}{k}$. We present the scale and aspect ratio distribution of our anchors with sliding window anchors in Figure 6. The results show great advantages of the guided anchoring scheme over predefined anchor scales and shapes. The predicted anchors cover a more wider range of scales and aspect ratios, which have a similar distribution to ground truth objects and provide a pool of initial anchors with higher coverage on ground-truth objects.

Feature adaption. The feature adaption module improves the proposal recall by a large margin, proving that a remedy of features consistency is essential. We claim that the improvement not only comes from adopting deformable convolution, but also results from our design of using anchor shape predictions to predict the offset of the deformable convolutional layer. If we simply add a deformable convolutional layer after anchor generation, the results of AR100/AR300/AR1000 are 56.1/62.4/66.1, which are inferior than our design.

The use of high-quality proposals. Despite with high-quality proposals, training a good detector is still a non-trivial problem. Adopting exactly the same settings can only bring limited gain. As illustrated in Figure 3, GA-RPN proposals provides more candidates of high IoU. This suggests that we can actually use fewer proposals for training a detector. To investigate this problem, we experiment with Fast R-CNN using RPN or GA-RPN proposals. We train the Fast R-CNN with different numbers of proposals, and adopt different IoU thresholds to assign labels for foreground/background.

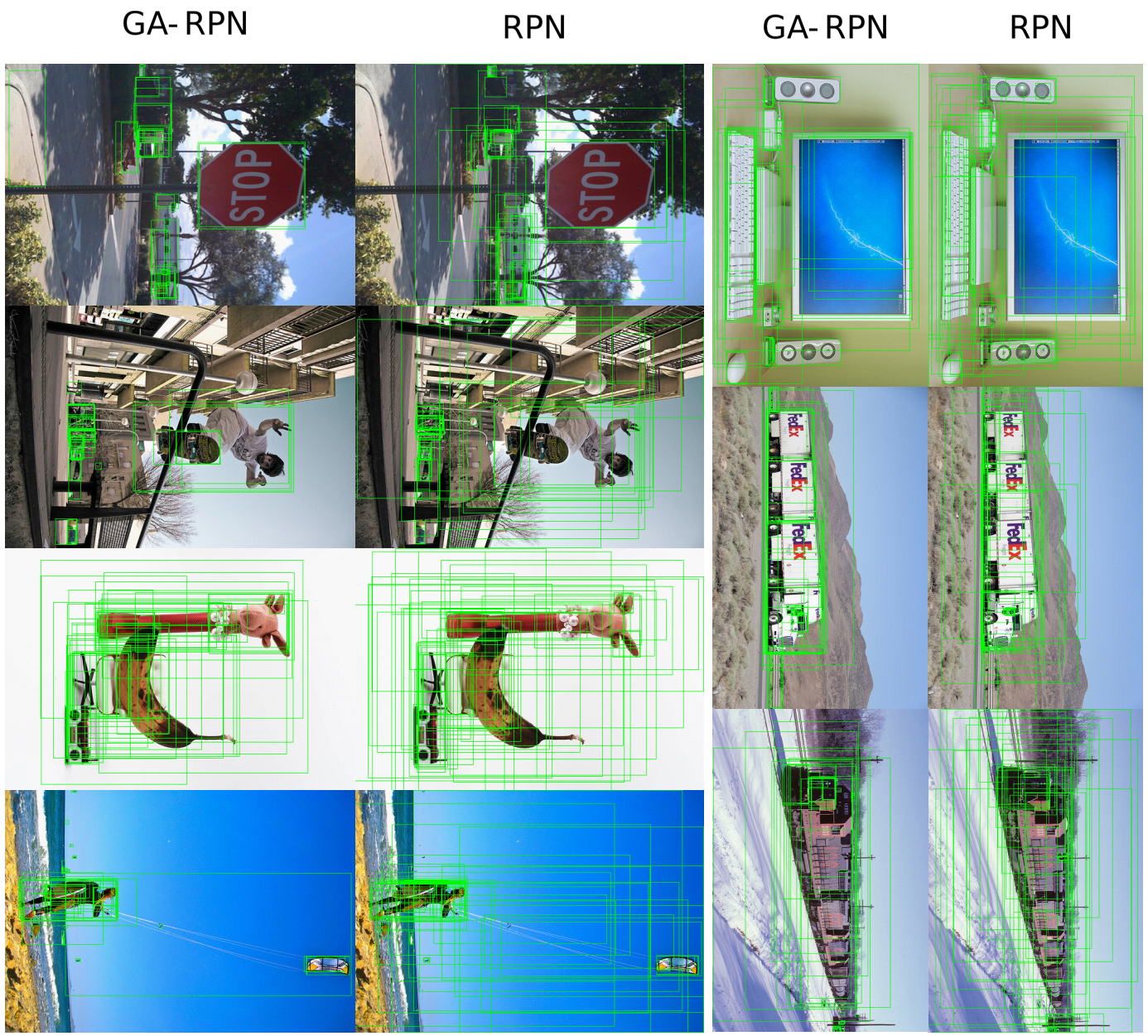
From the results in Table 6, we observe that: (1) Larger IoU thresholds is important for taking advantage of high-quality proposals. By concentrating on positive samples of higher IoU, there will be fewer false positives and the features for classification is more discriminative. Since we assign negative labels to proposals with IoU less than 0.6 during training, $AP_{0.5}$ will decrease and AP of high IoUs will increase by a large margin, and the total AP is much higher. (2) Using fewer proposals during training and testing can benefit the learning, in the condition of high recall. Fewer proposals lead to lower recall, but will simplify the learning process, since there are more hard samples in low-score proposals. When training with RPN proposals, the performance will decrease if we use only 300 proposals, because the recall is not sufficient and many objects get missed. However, GA-RPN guarantees high recall even with fewer proposals, thus training with 300 proposals could still boost the final mAP.

Table 6: Exploration of utilizing high-quality proposals.

proposal	num	IoU thr	AP	AP_{50}	AP_{75}
RPN	1000	0.5	36.7	58.8	39.3
	1000	0.6	37.2	57.1	40.5
	300	0.5	36.1	57.6	39.0
	300	0.6	37.0	56.3	39.5
GA-RPN	1000	0.5	37.4	**59.9**	40.0
	1000	0.6	38.9	59.0	42.4
	300	0.5	37.5	59.6	40.4
	300	0.6	**39.4**	59.3	**43.2**

5. Conclusion

We have proposed the Guided Anchoring scheme, which leverages semantic features to guide the anchoring. It generates nonuniform anchors of arbitrary shapes by jointly predicting the locations and anchor shapes dependent on locations. The proposed method achieved 9.1% higher recall with 90% fewer anchors than the RPN baseline using sliding window scheme. It can also be applied to various anchor-based detectors to improve the performance by as much as 2.7%.


GA- RPN
RPN
GA- RPN
RPN

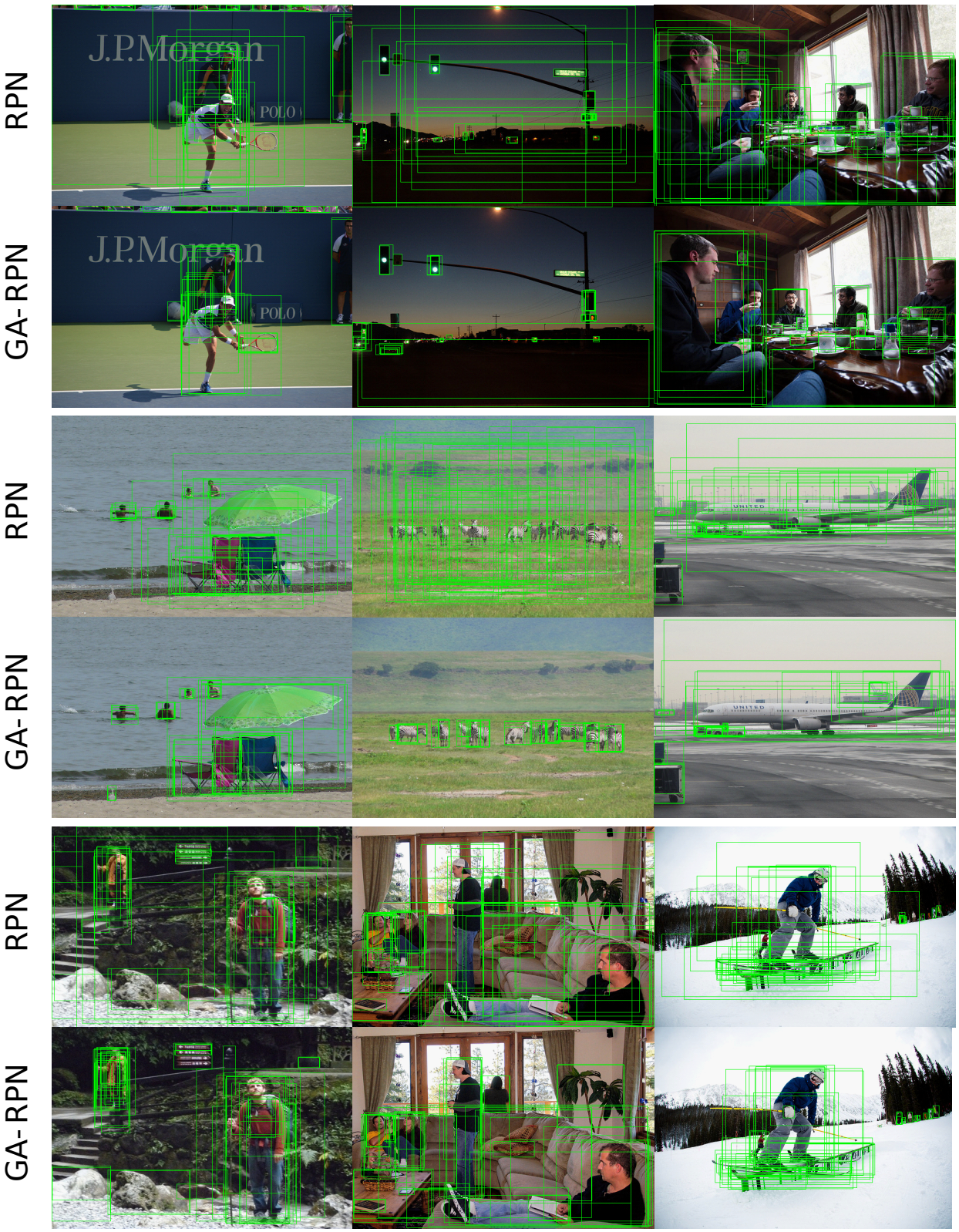


Figure 7: Examples of RPN proposals (top row) and GA-RPN proposals (bottom row).

References

[1] Z. Cai and N. Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 2

[2] J. Dai, Y. Li, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in Neural Information Processing Systems*, 2016. 2

[3] J. Dai, Y. Li, K. He, and J. Sun. R-FCN: Object detection via region-based fully convolutional networks. In *Advances in Neural Information Processing Systems*, 2016. 4

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005. 2

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009. 1

[6] P. Dollár, Z. Tu, P. Perona, and S. Belongie. Integral channel features. 2009. 2

[7] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, 2015. 1

[8] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012. 1

[9] A. Ghodrati, A. Diba, M. Pedersoli, T. Tuytelaars, and L. Van Gool. Deepproposal: Hunting objects by cascading deep convolutional layers. In *IEEE International Conference on Computer Vision*, 2015. 2

[10] S. Gidaris and N. Komodakis. Object detection via a multi-region and semantic segmentation-aware cnn model. In *IEEE International Conference on Computer Vision*, 2015. 2

[11] S. Gidaris and N. Komodakis. Attend refine repeat: Active box proposal generation via in-out localization. In *British Machine Vision Conference*, 2016. 7

[12] R. Girshick. Fast r-cnn. In *IEEE International Conference on Computer Vision*, 2015. 2, 6

[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014. 2

[14] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *IEEE International Conference on Computer Vision*, 2017. 2

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 6

[16] L. Huang, Y. Yang, Y. Deng, and Y. Yu. Densebox: Unifying landmark localization with end to end object detection. *arXiv preprint arXiv:1509.04874*, 2015. 2, 5

[17] Z. Jie, X. Liang, J. Feng, W. F. Lu, E. H. F. Tay, and S. Yan. Scale-aware pixelwise object proposal networks. *IEEE Transactions on Image Processing*, 25(10):4525–4539, 2016. 2

[18] H. Li, Y. Liu, W. Ouyang, and X. Wang. Zoom out-and-in network with map attention decision for region proposal and object detection. *International Journal of Computer Vision*, pages 1–14, 2017. 7

[19] X. Li, Z. Liu, P. Luo, C. C. Loy, and X. Tang. Not all pixels are equal: difficulty-aware semantic segmentation via deep layer cascade. 2017. 3

[20] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, July 2017. 2, 3, 6

[21] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *IEEE International Conference on Computer Vision*, 2017. 2, 3, 5, 7

[22] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, 2014. 1, 6

[23] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European Conference on Computer Vision*, 2016. 2

[24] D. G. Lowe. Object recognition from local scale-invariant features. In *IEEE International Conference on Computer Vision*. Ieee, 1999. 2

[25] H.-F. Lu, X. Du, and P.-L. Chang. Toward scale-invariance and position-sensitive region proposal networks. *European Conference on Computer Vision*, 2018. 7

[26] M. Najibi, M. Rastegari, and L. S. Davis. G-cnn: an iterative grid based object detector. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2369–2377, 2016. 2

[27] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollr. Learning to refine object segments. In *European Conference on Computer Vision*, 2016. 7

[28] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 2

[29] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 2

[30] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, 2015. 1, 2, 4, 5, 7

[31] G. Song, Y. Liu, M. Jiang, Y. Wang, J. Yan, and B. Leng. Beyond trade-off: Accelerate fcn-based face detector with higher accuracy. 2018. 3

[32] L. Tychsen-Smith and L. Petersson. Improving object localization with fitness nms and bounded iou loss. 2018. 5

[33] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171, 2013. 2

[34] X. Wu, D. Zhang, J. Zhu, and S. C. H. Hoi. Single-shot bidirectional pyramid networks for high-quality object detection, 2018. 2

[35] B. Yang, J. Yan, Z. Lei, and S. Z. Li. Craft objects from images. *arXiv preprint arXiv:1604.03239*, 2016. 2

[36] S. Zhang, L. Wen, X. Bian, Z. Lei, and S. Z. Li. Single-shot refinement neural network for object detection. *arXiv preprint arXiv:1711.06897*, 2017. 2, 6

[37] Q. Zhong, C. Li, Y. Zhang, D. Xie, S. Yang, and S. Pu. Cascade region proposal and global context for deep object detection. *arXiv preprint arXiv:1710.10749*, 2017. 2

[38] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *European Conference on Computer Vision*, 2014. 2