Augmenting Anchors by the Detector Itself

Xiaopei Wan[1*] , **Guoqiu Li**[2*] , **Yujiu Yang**[2] and **Zhenhua Guo**[3†]

[1]Ant Group
[2]Tsinghua University
[3]Alibaba Group

ranghou.wxp@alibaba-inc.com, lgq20@mails.tsinghua.edu.cn, yang.yujiu@sz.tsinghua.edu.cn,
cszguo@gmail.com

Abstract

Usually, it is difficult to determine the scale and aspect ratio of anchors for anchor-based object detection methods. Current state-of-the-art object detectors either determine anchor parameters according to objects' shape and scale in a dataset, or avoid this problem by utilizing anchor-free methods, however, the former scheme is dataset-specific and the latter methods could not get better performance than the former ones. In this paper, we propose a novel anchor augmentation method named AADI, which means **A**ugmenting **A**nchors by the **D**etector **I**tself. AADI is not an anchor-free method, instead, it can convert the scale and aspect ratio of anchors from a continuous space to a discrete space, which greatly alleviates the problem of anchors' designation. Furthermore, AADI is a learning-based anchor augmentation method, but it does not add any parameters or hyper-parameters, which is beneficial for research and downstream tasks. Extensive experiments on COCO dataset demonstrate the effectiveness of AADI, specifically, AADI achieves significant performance boosts on many state-of-the-art object detectors (eg. at least +2.4 box AP on Faster R-CNN, +2.2 box AP on Mask R-CNN, and +0.9 box AP on Cascade Mask R-CNN). We hope that this simple and cost-efficient method can be widely used in object detection. Code and models are available at https://github.com/WanXiaopei/aadi.

1 Introduction

Object detection, which needs to localize and recognize each object in images or videos, is a fundamental and critical problem in computer vision. With the development of deep learning, current object detectors are usually based on convolution neural networks, like FCOS [Tian *et al.*, 2019], RepPoints [Yang *et al.*, 2019; Chen *et al.*, 2020], TOOD [Feng *et al.*, 2021], Faster R-CNN[Ren *et al.*, 2016], DETR [Carion *et al.*, 2020], DeFCN [Wang *et al.*, 2021], and Sparse R-CNN [Sun *et al.*, 2021]. They can be roughly classified into
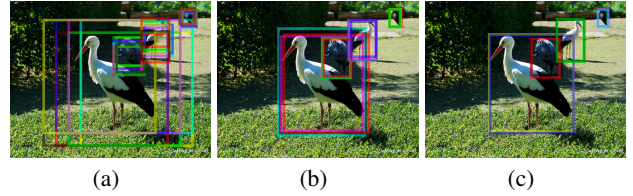

Figure 1: Visualization of AADI for RPN. (a) The original anchors of standard RPN. (b) The anchors augmented by AADI. (c) The proposals generated by AADI-RPN

anchor-free methods and anchor-based methods. Anchor-based methods firstly place a lot of anchors with different scales and aspect ratios, and then refine these anchors to generate proposals or detection results, while anchor-free methods directly generate detection results from feature maps.

The anchor-free methods were once widely regarded as the standard paradigm for future object detectors. However, ATSS [Zhang *et al.*, 2020] has demonstrated that anchor-free methods are not more effective than anchor-based ones. For example, with same tricks and appropriate strategies for judging positive and negative samples of hand-designed anchors, the anchor-based ATSS can perform better than anchor-free FCOS. Based on ATSS, PAA [Kim and Lee, 2020] proposes a method of judging positive and negative samples by Gaussian Mixture Model and achieves large improvements, which further proves the conclusion mentioned above. Both of these two models use a method to estimate the critical point of the distribution of positive and negative anchors. However, even if the best critical point is estimated, there is no guarantee that the positive and negative samples will be completely separated, and placing more hand-designed anchors can not solve this problem. Therefore, how to improve the quality of anchors has become an important problem.[1]

In current anchor-based object detectors' paradigm, high-quality anchors usually lead to better performance, because they make it easier to separate positive and negative anchors, which is important to improve the recall rate of ground truth boxes and reduce false positives. In standard RPN, anchors with an IoU greater than 0.7 are marked as positives, those with an IoU below 0.3 are marked as negatives, and others

---

*These authors have contributed equally to this work.
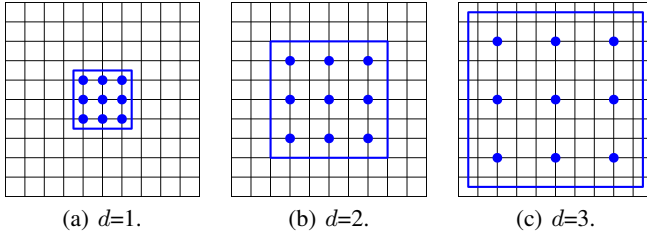†Corresponding author.

[1]More details can be found in https://arxiv.org/abs/2105.14086

Figure 2: Anchors under different dilations, and the kernel of the convolution layer is fixed to 3×3.

are ignored in training. Thus, placing more anchors with different aspect ratios and scales will improve the performance of RPN. However, since the aspect ratio and scale can be any positive number, it makes the anchor designation a non-trivial problem. Thus, some methods try to learn high-quality anchors by a neural network module, like AttratioNet [Gidaris and Komodakis, 2016], GA-RPN, and Cascade RPN. All these methods can be viewed as gradient-based anchor augmentation methods, and their overall training and inference overheads almost double due to their strategy of dense prediction and large memory accesses in feature alignment.

In this paper, we propose a gradient-free anchor augmentation method named AADI, which means **A**ugmenting **A**nchors by the **D**etector **I**tself. AADI can be applied to both two-stage and single-stage methods, and Figure 1 shows an example of AADI for RPN [Ren *et al.*, 2016]. AADI comprises two processes, including augmentation and refinement process. During training process, AADI first uses RPN to augment hand-designed anchors to improve their quality in augmentation process, and then uses the augmented anchors and ground truth to train the parameters of RPN in refinement process. Intuitively, AADI can be viewed as a variational EM algorithm which is typically used to solve the problems with latent variables, and the augmentation and refinement process are corresponding to its E-step and M-step, respectively. Here, we iterate it only once to improve the computational efficiency. Moreover, in the augmentation process, AADI uses dilated convolution [Yu and Koltun, 2015] to augment each hand-designed anchor, and in the refinement process, it employs RoI Align [He *et al.*, 2017] to extract features for augmented anchors and uses an equivalent fully connected layer to refine these anchors. Note that the dilated convolution and fully connected operations share parameters.

In order to make RPN's parameters be reused in these two processes, we need to carefully design the dilation and kernel size of convolution layer to ensure that the convolution layer and fully connected layer could obtain same input feature for a specific anchor. As Figure 2 shows, supposing that the dilation of a convolution layer is d, then the process of a $3{\times}3$ convolution layer is same as that of an RoI Align for a $(3{\times}d){\times}(3{\times}d)$ anchor. Furthermore, the operation of a $3{\times}3$ convolution layer for each anchor is equal to a fully connected layer whose output dimension is 9. Thus, unlike the ineffective masked deformable convolution layer in GA-RPN or adaptive convolution layer in Cascade RPN which is used for tackling the feature misalignment problem of hand-designed

and learned anchors, AADI avoids this problem by determining the scale and aspect ratio of anchors according to the dilation and kernel size of convolution operation. Consequently, the hyper-parameters of AADI are kernel size and dilation of convolution layer, that is, AADI converts the scale and aspect ratio of anchors from a continuous space to a discrete space, which greatly alleviates the problem of anchor designation.

During training, the augmentation process does not need to calculate gradient, which makes the training speed of AADI fast. During inference, AADI only takes top 2000 augmented anchors for further refinement, which makes the inference speed fast. Extensive experiments on MS COCO dataset show that AADI can efficiently improve the detection performance of anchor-based object detectors, specifically, AADI achieves at least 2.4 box AP improvements on Faster R-CNN, 2.2 box AP improvements on Mask R-CNN, 1.8 box AP improvements on RetinaNet, and 0.9 box AP improvements on Cascade Mask R-CNN. And it runs only slightly slower than standard RPN. Our main contributions are as follows:

- We propose a novel and effective method, AADI, which augments anchors by the detector itself and can significantly improve anchors' quality. It can be applied on all anchor-based object detectors without adding any parameters or hyper-parameters.

- AADI determines the scale and aspect ratio of anchors according to the kernel and dilation of a convolution layer, and it determines the aspect ratio according to the shape of the kernel, which greatly eases the anchor design difficulty.

- We carry out experiments on MS COCO dataset to verify the effectiveness of AADI. With only little extra computation costs, AADI achieves significant performance improvements on many representative object detectors with different backbones.

2 Related Works

Current object detection methods can be broadly categorized into anchor-based methods and anchor-free methods. Anchor-based methods firstly place many hand-designed anchors on the feature maps, and then perform classification and box regression on these anchors. Therefore, better anchors can greatly alleviate the difficulty of these two tasks. AttratioNet [Gidaris and Komodakis, 2016] uses ARN to iteratively refine the hand-designed anchors several times. However, ARN needs to employ RoI Pooling to extract features for each anchor, and it is not a lightweight network, which makes its training and inference speed slow. GA-RPN [Wang *et al.*, 2019] first predicts a suitable anchor for each position, and then uses the predicted anchors to train RPN. Cascade RPN [Cai and Vasconcelos, 2018] stacks two RPNs together to improve the quality of proposals. Its first RPN is used to refine the hand-designed anchors without classification, and the second one plays the same role as a standard RPN. GA-RPN and Cascade RPN have effectively improved the quality of proposals by improving the quality of anchors. However, they both need to use DCN[Dai *et al.*, 2017] or its variants to align the features and anchors, and there is a lot of memory access during this operation, resulting in slower training

Algorithm 1 Pseudocode of AADI

Input:
    $\mathcal{A}$ is a set of hand-designed anchors;
    $\mathcal{A}_a$ is a set of augmented anchors;
    $\mathcal{G}$ is a set of ground truth boxes on the image;
    $\mathcal{F}$ is feature maps obtained by backbone;
    f is RPN;
    θ is a set of paramters of RPN;
    ϵ is learning rate;
Output:
    $\mathcal{P}$ is a set of proposals;
1: **while** not converge **do**
2:    $\mathcal{A}_a \leftarrow f(\mathcal{A}, \mathcal{F}, \theta)$
3:    $\mathcal{P} \leftarrow f(\mathcal{A}_a, \mathcal{F}, \theta)$
4:    $\mathcal{L} \leftarrow loss(\mathcal{P}, \mathcal{G})$
5:    $\triangle_\theta \leftarrow \frac{\partial \mathcal{L}}{\partial f(\mathcal{A}_a, \mathcal{F}, \theta)} \times \frac{f(\mathcal{A}_a, \mathcal{F}, \theta)}{\partial \theta}$
6:    $\theta \leftarrow \theta - \epsilon \times \triangle_\theta$
7: **end while**

and inference speed. In this paper, we first filter a small set of augmented anchors, and then use RoI Align to extract the features for augmented anchors, which can significantly reduce the memory access.

Compared with anchor-free methods, anchor-based methods are more intuitive, and they do not need to determine which ground truth should be placed on which level of pyramid features. However, FCOS [Tian *et al.*, 2019] finds that determining the scale and aspect ratio of anchors is not a trivial issue, and these two parameters have a great influence on the performance. Thus, FCOS proposes an anchor-free paradigm to generate the final results. For each position which is located in the center region of the ground truth, it directly predicts the distance from the position to the four corners of the ground truth. RepPoints [Yang *et al.*, 2019] is also an anchor-free method. It no longer directly predicts bounding boxes, but predicts a set of representative points for each ground truth, and then groups them to bounding boxes. DETR and Deformable DETR [Zhu *et al.*, 2020] use Transformer to directly predict the detection results to avoid the anchor designation problem.

Although the hyper-parameter adjustment of anchor-free methods is simpler, ATSS and PAA demonstrate that anchor-based methods can significantly outperform anchor-free methods with suitable anchor assignment strategies. However, the anchors of both ATSS and PAA are predefined, although both ATSS and PAA are very robust to the scale and aspect ratio of anchors, they are hard to further improve the performance by placing more anchors with different scales and aspect ratios. However, AADI can adaptively augment the hand-designed anchors to improve their quality.

3 Method

3.1 AADI

High-quality anchors are critical for all anchor-based object detectors, so, how to adaptively generate better anchors is an important factor. Unlike previous gradient-based methods, AADI just uses the detector itself to augment the hand-

designed anchors without gradient, and it can be used in both two-stage and single-stage object detectors. The overall pipeline of AADI for a common two-stage object detector is shown in Figure 3(b), and its Pseudocode is shown in Algorithm 1. Compared with AttratioNet, AADI does not need to calculate gradient during anchor augmentation. And compared with the IBBR(**I**terative **B**ounding **B**ox **R**egression) of IoU-Net [Jiang *et al.*, 2018], which has a similar inference pipeline. The key difference is that AADI works on both training and inference period, while IBBR only works on inference. Taking AADI for two-stage object detectors as an example, AADI comprises two processes, including an augmentation process and a refinement process, and the details are illustrated as follows.

Augmentation process. As shown in Figure 3(b), AADI first gets the input of anchors and feature maps from the backbone, and then uses RPN_c to augment the anchors. RPN_c is same as the standard RPN, and it does not need to calculate the gradient during training. As most of augmented anchors are either redundant samples or negative samples, so we remove them by Non-Maximum Suppression (NMS) and keep only 2000 anchors with the highest score to improve the model's efficiency. As shown in Figure 1, the augmented anchors have a higher Intersection over Union (IoU) than original anchors.

Note that there can be multiple RPN_c with different dilations, and these RPN_cs can share parameters.

Refinement process. The refinement process gets input of augmented anchors from the augmentation process and feature maps from the backbone, and then uses RPN_f to predict proposals. RPN_f and RPN_c have the same parameters, but they are implemented in different ways. RPN_c is same as the standard RPN, and it uses convolution to get augmented anchors, while RPN_f is implemented in a fully connected way. In addition, RPN_f employs RoI Align to extract features for each augmented anchor. Note that different RPN_cs are corresponding to different RPN_fs if their parameters are not sharing. Moreover, RPN_f needs to calculate gradient during training, so RPN_c will also be trained indirectly.

The key problem of ensuring that this paradigm can work is the features obtained by the two RPNs should be same. For RPN_c, the features for each anchor are determined by the kernel size and dilation of a convolution layer, while for RPN_f, the features are determined by RoI Align. So, there are two conditions to meet. First, the output size of RoI Align needs to be same as the kernel size of the convolution layer, which is easy to meet. Second, the dilation and the scale of each anchor must be carefully designed to align the RoI, which is hard to meet.

To address this issue, we propose to set scale and aspect ratio of hand-designed anchors by Eq. 1 and Eq. 2, respectively. Where m and n are height and width of kernel size of a convolution layer, and d is the dilation. In Figure 2, we visualize the anchors and their features sampled by a 3×3 convolution layer under different dilations, and it is easy to find that the features sampled by a 3×3 convolution layer are same as the features extracted by a 3×3 RoI Align. In this way, the aspect ratio of the anchors is also determined by the shape of the ker-

(a) RPN.

(b) AADI-RPN.

(c) Box subnet of RetinaNet.
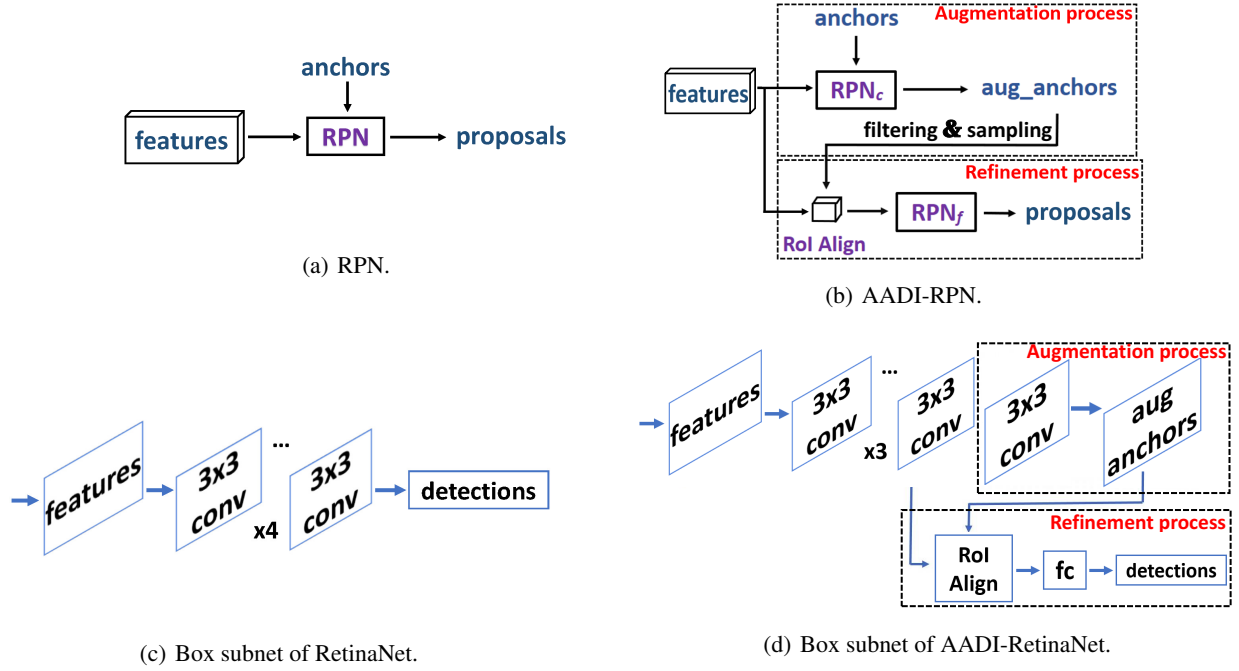
(d) Box subnet of AADI-RetinaNet.

Figure 3: The pipeline of AADI for RPN and RetinaNet.

nel. For example, for a 3×3 convolution layer, if its dilation is set to 2, then the scale of the anchors is $3 \times 2 = 6$, and the aspect ratio is $\frac{3}{3} = 1$. Since m, n and d all are must be integers, so the designation space of anchors is converted from a continuous space to a discrete space, which greatly alleviates the problem of anchor designation.

$$scale = d \times \sqrt{m \times n} \qquad (1)$$

$$aspect\_ratio = \frac{n}{m} \qquad (2)$$

3.2 Training

If AADI is only trained on the augmented anchors, AADI's robustness to hand-designed anchors is relatively poor, which will reduce the performance of anchor augmentation. Therefore, during training, we first select an anchor with the largest IoU for each ground truth, then append them to the augmented anchors, which can guide the AADI to augment the hand-designed anchors, we call this strategy **Anchor Guided**.

The number of positive augmented anchors is very large, if all of them are used directly, the memory will be exhausted. In addition, we found that the overlap of some augmented anchors is large. Therefore, we employ NMS to filter out the redundant anchors according to their IoUs with the ground truth, and the training strategy of the refinement process is same as standard RPN. Similar to RPN, AADI can be trained in an end-to-end manner using multi-task loss as follows:

$$\mathcal{L}_{rpn} = \lambda \mathcal{L}_{reg} + \mathcal{L}_{cls} \qquad (3)$$

Where $\mathcal{L}_{reg}$ and $\mathcal{L}_{cls}$ are the box regression loss and the classification loss, respectively. And these two loss terms are balanced by λ. In the implementation, binary cross entropy loss and Smooth L1 Loss [Girshick, 2015] are used as classification loss and box regression loss, respectively, and λ is empirically set to 5.

3.3 Inference

During inference, the augmentation process is same as that of obtaining proposals by standard RPN, and the refinement process is just a two-layer fully connected layer which is used to refine the augmented anchors. Here, AADI only takes the top 2000 augmented anchors for refinement, while Cascade RPN needs to augment all regressed anchors, which leads to higher computation overheads.

3.4 AADI-RetinaNet

AADI can be naturally deployed in anchor-based single-stage object detectors. We only apply AADI on RetinaNet (AADI-RetinaNet) due to the limited time and resources. Since RetinaNet uses 4 multi-layer convolution networks to extract features for classification and regression tasks respectively, and the kernel of all convolution layers is 3×3, so we can use the last layer of the box subnet to extract features for the augmented anchors. The pipeline of AADI-ReitinaNet are shown in Figure 3(d), and the training process of AADI-RetinaNet is similar to AADI-RPN, However, during inference, since NMS is relatively time-consuming, we only select the top 2000 augmented anchors without NMS to do further refinement. Here, the IoU threshold of NMS of post-processing is set to 0.6. Furthermore, Group Normalization [Wu and He, 2018] is used to make the training process smoother.

Method	Backbone	AR_{100}	AR_s	AR_m	AR_l	AP^{box}	Speed Train(ms)	Speed Test(FPS)
AttratioNet	VGG-16	53.3	31.5	62.2	77.7	34.3	-	-
ZIP	Inception	53.9	31.9	63.0	78.5	-	-	-
C-RPN	R-50	54.0	35.6	62.7	78.5	-	-	-
RPN	R-50	45.7	31.4	52.7	61.1	37.9	186.0	31.3
PAA	R-50	49.7	31.4	59.4	68.3	39.3	342.6	31.4
GA-RPN	R-50	59.1	40.7	68.2	78.4	39.6	448.1	14.7
Cascade RPN	R-50	61.1	42.1	69.3	82.8	40.4	436.4	15.1
AADI-RPN	R-50	55.3	35.4	62.7	79.7	41.3	170.2	25.6

Table 1: Region proposal results on COCO *val-2017*. All results are trained with 1x schedule.

4 Experiments

4.1 Dataset and Evaluate Metrics

Our experiments are implemented on the challenging Microsoft COCO 2017 [Lin *et al.*, 2014] dataset. It consists of 118k images for training, *train-2017* and 5k images for validation, *val-2017*. There are also 20k images without annotations for test in *test-dev*.

We train all our models on *train-2017*, and report ablation studies on *val-2017* and final results on *test-dev*. In this paper, Average Recall(AR) is used to measure the quality of proposals, which is the average of recalls at different IoU thresholds, and AR_x means recall at IoU threshold of $x\%$, AR_s, AR_m, and AR_l denote the AR for small, medium, and large objects, respectively. The detection results are subjected to standard COCO-style Average Precision (AP) metrics which include AP (mean of AP over all IoU thresholds), AP_{50}, AP_{75}, AP_s, AP_m and AP_l, where AP_{50} and AP_{75} are APs for IoU threshold of 50% and 75%, respectively, and AP_s, AP_m and AP_l are APs for small, medium, and large objects, respectively. Moreover, AP^{box} and AP^{mask} are denoted as AP for object detection and instance segmentation, respectively.

4.2 Implementation Details

All experiments are based on detectron2 [Wu *et al.*, 2019]. And our codes are deployed on a machine with 8 Tesla V100-SXM2-16GB GPUs and Intel Xeon Platinum 8163 CPU. Our software environment mainly includes Ubuntu 18.04 LTS OS, CUDA 10.1, and PyTorch [Paszke *et al.*, 2017] 1.6.0.

The hyper-parameters of our model are following detectron2, and all models are based on FPN [Lin *et al.*, 2017]. For AADI-RPN, the dilations of RPN_c is set to 2 and 4, and these two RPN_c are not sharing parameters, moreover. For AADI-RetinaNet, the dilation is set to 2. Moreover, the strategy of Anchor Guided is used in all models.

As a common practice [Girshick *et al.*, 2014], all network backbones are pretrained on ImageNet1k classification set [Deng *et al.*, 2009], and their pre-trained models are publicly available. With 1x [Wu *et al.*, 2019] training schedule and ResNet-50 (R-50) as backbone, it takes about 4.2 hours for our model to converge, and about 2.9 hours with Mixed Precision training [Micikevicius *et al.*, 2017].

4.3 Results

Region Proposal Performance. We first apply AADI on RPN (AADI-RPN) of Faster R-CNN to verify its perfor-

mance on region proposal, and the model is trained on COCO *train-2017* with 1x training schedule. Since it is difficult to reproduce the performance of AttratioNet, ZIP[Li *et al.*, 2019] and C-RPN [Fan and Ling, 2019], we only compare recall rate with published results. For other models, their performance is tested on the same machine, and the training time indicates the average time cost per batch, while the test time indicates the average inference speed on COCO *val-2017*. All region proposal results are shown in Table 1, and the results show that AADI-RPN, by a large margin, outperforms standard RPN in terms of AR under all settings, and it achieves the highest AP^{box} over other models. Furthermore, although AADI-RPN has a lower recall rate than the current state-of-the-art Cascade RPN, its inference speed is about 70% faster. For training time, AADI-RPN achieves the fastest training speed, even faster than RPN. This is mainly because only a small set of the augmented anchors are participated in the training process, and the way to compute the gradient is changed from convolution to fully connected operation, which significantly reduces the training overheads.

Object Detection Performance. We conducted extensive experiments on four typical object detection frameworks: RetinaNet, Faster R-CNN, Mask R-CNN, and Cascade Mask R-CNN, as well as two different backbone networks: R-50 and tiny Swin Transformer (Swin-T) [Liu *et al.*, 2021] to verify the anchor augmentation capabilities of AADI. All detection results with 3x training schedule are shown in Table 2, and we can see that AADI achieves significant performance boosts in all detectors with different backbones, which demonstrates its strong robustness. In particular, when using Swin-T as the backbone, we observe +2.1 AP^{box}, +2.7 AP^{box}, +2.4 AP^{box}, and +0.9 AP^{box} gains over RetinaNet, Faster R-CNN, Mask R-CNN, and Cascade Mask R-CNN, respectively. Furthermore, their AP^{mask} are also improved. It is worth noting that the Swin Transformer based object detectors are currently state-of-the-art, while AADI still achieves significant performance boosts with only little extra computation costs, which reflects the efficiency of AADI. Meanwhile, we can find that the performance gains on AP_{50}^{box} are small, while large gains are obtained on AP_{75}^{box}, similar finding could be found for GA-RPN and Cascade RPN. Therefore, the main advantage of AADI is to improve the quality of anchors, thereby improving the detection performance.

5 Ablation Study

We conduct our ablation study on COCO *train-2017*, and report the results on COCO *val-2017* dataset. All models' backbone is R-50 unless specially noted.

It can be seen from Eq. 1 and Eq. 2 that the scale and aspect ratio of anchors is determined by the kernel and dilation of a convolution layer. For RPN, since the kernel is usually fixed to 3×3, so we only change the scale of anchors by adjusting the dilation. Here, we test the region proposal performance of RPN with dilation of 2, 3 and 4, respectively. Furthermore, the effects of the proposed training strategy of **Anchor Guided** in augmentation process are also tested. All results are shown in Table 3.

From Table 3, we can conclude that larger dilation, worse

Method	Backbone	AADI	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}	Inference Speed(FPS)
RetinaNet	R-50		39.6	59.3	42.2	-	-	-	22.2
	R-50	✓	**41.4**	**59.3**	**45.2**	-	-	-	23.0
	Swin-T		44.9	**65.8**	48.0	-	-	-	18.6
	Swin-T	✓	**47.0**	65.6	**51.4**	-	-		18.1
Faster R-CNN	R-50		40.4	61.4	43.9	-	-	-	26.0
	R-50	✓	**42.8**	**61.4**	**46.9**	-	-	-	18.7
	Swin-T		45.0	67.2	49.1	-	-	-	19.6
	Swin-T	✓	**47.7**	**67.5**	**52.4**	-	-	-	15.1
Mask R-CNN	R-50		41.0	61.5	44.9	37.2	58.6	39.9	22.8
	R-50	✓	**43.2**	**61.7**	**47.7**	**38.1**	**58.9**	**41.2**	15.8
	Swin-T		46.0	**68.1**	50.3	41.6	**65.1**	44.9	17.9
	Swin-T	✓	**48.4**	67.6	**53.2**	**42.8**	65.0	**46.6**	13.4
Cascade Mask R-CNN	R-50		44.4	61.7	48.3	38.2	59.0	41.1	12.5
	R-50	✓	**45.4**	**62.4**	**49.2**	**38.9**	**59.4**	**42.0**	9.9
	Swin-T		50.3	69.0	54.9	44.0	66.5	47.8	10.7
	Swin-T	✓	**51.2**	**69.2**	**55.8**	**44.1**	**66.6**	**48.1**	8.9

Table 2: Object detection results on COCO *val-2017*. All results are trained with 3x schedule.

Dilation	Anchor Guided	AR_{100}	AR_{1000}	AR_s	AR_m	AR_l
2		54.8	64.7	39.0	63.1	70.6
2	✓	56.3	66.7	39.5	64.9	73.4
3		53.7	64.0	35.4	62.1	73.9
3	✓	55.6	67.6	36.1	64.3	77.6
4		52.2	60.5	30.9	61.3	76.6
4	✓	54.4	65.5	33.0	63.7	78.9
RPN		45.7	58.0	31.4	52.7	61.1

Table 3: Region proposal results of AADI-RPN with different settings. All results are trained with 1x schedule.

Dilation	RPN Parameter	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
2	-	40.3	59.3	44.3	24.2	43.3	52.2
3	-	40.8	59.5	45.0	24.0	44.6	53.1
4	-	40.5	58.7	44.6	23.2	44.8	52.7
2,3	share	40.8	59.6	45.2	24.4	44.0	53.0
3,4	share	40.7	59.2	44.5	23.3	44.6	54.0
2,4	share	40.6	59.0	44.5	23.7	44.0	52.4
2,3	not share	40.7	59.5	44.7	**24.5**	44.1	52.8
3,4	not share	41.2	59.7	**46.0**	23.7	45.0	54.0
2,4	not share	**41.3**	**59.9**	45.5	23.9	**45.4**	**54.0**
2,3,4	not share	41.0	59.6	45.1	24.4	44.7	53.6
Faster R-CNN	-	37.9	58.8	41.1	22.4	41.1	49.1

Table 4: Detection performance of AADI for Faster R-CNN with different dilations. All results are trained with 1x schedule.

performance of small objects, and better performance of large objects, and vice versa. This is because the scale of anchors will become larger as the dilation becomes larger, and large anchors are beneficial for large objects, but harmful for small objects during training. Furthermore, the strategy of Anchor Guided can consistently improve the performance of AADI-RPN with different dilations, especially for large objects. Compared with standard RPN, AADI-RPN consistently gets better performance by a large margin under different settings, which indicates that AADI is an effective method.

We find that there is not much difference in the performance of AADI-RPN with different dilations. Therefore, we test detection performance to determine which dilation or its combination is the optimal choice. We further test whether the parameters are shared or not when using different dilation combinations. All detection results are shown in Table 4, and the strategy of Anchor Guided is used for all tests.

From Table 4, we empirically find that setting dilation to 3 (the scale of anchors is 9) achieves the best detection performance when only one dilation is used, which may correspond to the dominate scale of objects in the dataset. We also observe a small performance gap among different dilations, which indicates the robustness of AADI. Moreover, since different dilations correspond to different scales of anchors, we expect that using two or more RPNs with different dilations will bring further performance gains. Therefore, we deploy two or more RPNs with different dilations in AADI and test the performance of these different dilation combinations. The results show that when two RPNs share the same parameters, there is almost no performance gain compared to using only one RPN. When the parameters are not shared, the perfor-

mance can be substantially improved, and we can obtain the highest AP of 41.3 when the dilations of two RPNs are set to 2 and 4, respectively. On top of this, we add another RPN with dilation of 3 to test whether the performance can be further improved. However, a lower AP of 41.0 is obtained, probably due to newly added RPN parameters raising the difficulty of optimization. Moreover, compared with Faster R-CNN, our best practice of AADI achieves 3.4 AP gains.

6 Conclusion

In this paper, we propose a gradient-free anchor augmentation method named AADI. It does not add any parameters or hyper-parameters, and it does not significantly affect training and inference speed. We conduct experiments on RetinaNet and different R-CNNs to verify its performance. Specifically, AADI brings at least 2.2 box AP improvements on Faster/-Mask R-CNN, 1.8 box AP improvements on RetinaNet, and 0.9 box AP gains on Cascade Mask R-CNN. In our future work, we will try to combine AADI and other anchor-based methods together to further improve detection performance.

Acknowledgements

References

[Cai and Vasconcelos, 2018] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *CVPR*, 2018.

[Carion *et al.*, 2020] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*. Springer, 2020.

[Chen *et al.*, 2020] Yihong Chen, Zheng Zhang, Yue Cao, Liwei Wang, Stephen Lin, and Han Hu. Reppoints v2: Verification meets regression for object detection. In *NeurIPS*, 2020.

[Dai *et al.*, 2017] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *ICCV*, 2017.

[Deng *et al.*, 2009] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009.

[Fan and Ling, 2019] Heng Fan and Haibin Ling. Siamese cascaded region proposal networks for real-time visual tracking. In *CVPR*, 2019.

[Feng *et al.*, 2021] Chengjian Feng, Yujie Zhong, Yu Gao, Matthew R Scott, and Weilin Huang. Tood: Task-aligned one-stage object detection. In *ICCV*, 2021.

[Gidaris and Komodakis, 2016] Spyros Gidaris and Nikos Komodakis. Attend refine repeat: Active box proposal generation via in-out localization. *arXiv preprint arXiv:1606.04446*, 2016.

[Girshick *et al.*, 2014] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[Girshick, 2015] Ross Girshick. Fast r-cnn. In *ICCV*, 2015.

[He *et al.*, 2017] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017.

[Jiang *et al.*, 2018] Borui Jiang, Ruixuan Luo, Jiayuan Mao, Tete Xiao, and Yuning Jiang. Acquisition of localization confidence for accurate object detection. In *ECCV*, 2018.

[Kim and Lee, 2020] Kang Kim and Hee Seok Lee. Probabilistic anchor assignment with iou prediction for object detection. In *ECCV*, 2020.

[Li *et al.*, 2019] Hongyang Li, Yu Liu, Wanli Ouyang, and Xiaogang Wang. Zoom out-and-in network with map attention decision for region proposal and object detection. *International Journal of Computer Vision*, 127(3):225–238, 2019.

[Lin *et al.*, 2014] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*. Springer, 2014.

[Lin *et al.*, 2017] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[Liu *et al.*, 2021] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021.

[Micikevicius *et al.*, 2017] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.

[Paszke *et al.*, 2017] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[Ren *et al.*, 2016] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2016.

[Sun *et al.*, 2021] Peize Sun, Rufeng Zhang, Yi Jiang, Tao Kong, Chenfeng Xu, Wei Zhan, Masayoshi Tomizuka, Lei Li, Zehuan Yuan, Changhu Wang, et al. Sparse r-cnn: End-to-end object detection with learnable proposals. In *CVPR*, 2021.

[Tian *et al.*, 2019] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *ICCV*, 2019.

[Wang *et al.*, 2019] Jiaqi Wang, Kai Chen, Shuo Yang, Chen Change Loy, and Dahua Lin. Region proposal by guided anchoring. In *CVPR*, 2019.

[Wang *et al.*, 2021] Jianfeng Wang, Lin Song, Zeming Li, Hongbin Sun, Jian Sun, and Nanning Zheng. End-to-end object detection with fully convolutional network. In *CVPR*, 2021.

[Wu and He, 2018] Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, 2018.

[Wu *et al.*, 2019] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

[Yang *et al.*, 2019] Ze Yang, Shaohui Liu, Han Hu, Liwei Wang, and Stephen Lin. Reppoints: Point set representation for object detection. In *ICCV*, 2019.

[Yu and Koltun, 2015] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[Zhang *et al.*, 2020] Shifeng Zhang, Cheng Chi, Yongqiang Yao, Zhen Lei, and Stan Z Li. Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In *CVPR*, 2020.

[Zhu *et al.*, 2020] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.