Augmenting Anchors by the Detector Itself

Xiaopei Wan1∗, Guoqiu Li2∗, Yujiu Yang2 and Zhenhua Guo3†
1Ant Group
2Tsinghua University
3Alibaba Group
ranghou.wxp@alibaba-inc.com, lgq20@mails.tsinghua.edu.cn, yang.yujiu@sz.tsinghua.edu.cn, csguo@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 Augmenting Anchors by the Detector Itself. 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 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

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

1More details can be found in https://arxiv.org/abs/2105.14086
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 augmentation methods, and their overall training and 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 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.

![Figure 2: Anchors under different dilations, and the kernel of the convolution layer is fixed to $3 \times 3$.](image-url)
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;
- $\theta$ is a set of parameters of RPN;
- $\epsilon$ 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: $\mathcal{L} \leftarrow \text{loss}(\mathcal{P}, \mathcal{G})$
5: $\delta \theta \leftarrow \frac{\partial \mathcal{L}}{\partial \mathcal{A}_a} \mathcal{F} \theta \mathcal{G} \times \frac{f(\mathcal{A}_a, \mathcal{F}, \theta)}{\partial \theta}$
6: $\theta \leftarrow \theta - \epsilon \times \delta \theta$
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 AttrationNet, AADI does not need to calculate gradient during anchor augmentation. And compared with the IBBR (Iterative Bounding Box Regression) 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 RPNc to augment the anchors. RPNc 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 the original anchors.

Note that there can be multiple RPNc with different dilations, and these RPNc’s 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 RPNf to predict proposals. RPNf and RPNc have the same parameters, but they are implemented in different ways. RPNc is same as the standard RPN, and it uses convolution to get augmented anchors, while RPNf is implemented in a fully connected way. In addition, RPNf employs RoI Align to extract features for each augmented anchor. Note that different RPNc’s are corresponding to different RPNf’s if their parameters are not sharing. Moreover, RPNf needs to calculate gradient during training, so RPNf 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 RPNc, the features for each anchor are determined by the kernel size and dilatation of a convolution layer, while for RPNf, 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 dilatation 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 dilatation. In Figure 2, we visualize the anchors and their features sampled by a $3 \times 3$ convolution layer under different dilations, and it is easy to find that the features sampled by a $3 \times 3$ convolution layer are same as the features extracted by a $3 \times 3$ RoI Align. In this way, the aspect ratio of the anchors is also determined by the shape of the ker-
nel. For example, for a $3 \times 3$ 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 $\frac{3}{2} = 1$. 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.

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

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

### 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:

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

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 $\lambda$. In the implementation, binary cross entropy loss and Smooth L1 Loss [Girshick, 2015] are used as classification loss and box regression loss, respectively, and $\lambda$ 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 $3 \times 3$, so we can use the last layer of the box subnet to extract features for the augmented anchors. The pipeline of AADI-RetinaNet 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.
Table 1: Region proposal results on COCO val-2017. All results are trained with 1x schedule.

| Method         | Backbone | AR_{100} | AR_{50} | AR_{m} | AR_{s} | AP^{box} | Speed |
|----------------|----------|----------|---------|--------|--------|----------|-------|
| AttratioNet    | VGG-16   | 55.3     | 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 |
| FFA           | 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 |

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 20k 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_{s} means recall at IoU threshold of 50%, AR_{m}, and AR_{s} 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}, AR_{s}, AR_{m}, and AR_{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_{s} is set to 2 and 4, and these two RPN_{s} 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.

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 x 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...
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 performance 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

This work was partially supported by The Key Program of National Natural Science Foundation of China (No. U1903213), the Natural Science Foundation of China (No. 61772296) and Shenzhen Fundamental Research Fund (No. JCYJ20170412170438636).
