Supplementary Material for “Robustizing Object Detection Networks Using Augmented Feature Pooling”

Takashi Shibata[1], Masayuki Tanaka[2], and Masatoshi Okutomi[2]

[1] NTT Corporation, Kanagawa, Japan,
[2] Tokyo Institute of Technology, Tokyo, Japan

Abstract. This is the supplementary material for “Robustizing Object Detection Networks Using Augmented Feature Pooling.” We provide additional experimental results, a pseudo-code for our algorithm and details of our training protocols, and the details of datasets used in our experiments.

1 Additional Results

In this section, we report additional results for (i) applicability to various backbones, (ii) comparison with data augmentation (DA) and test time augmentation (TTA), and (iii) applicability to modern object detection architectures.

Applicability to Various Backbones (Sec. 5.2). We demonstrate the effectiveness of our proposed method in terms of applicability to various backbones in Sec. 5.2. Figure 1 shows additional visual comparisons. These results also support the superiority of the proposed method over vanilla, as in our main paper. More example results by our proposed method are shown in Fig. 2. Our proposed method successfully detects those objects containing various rotations[3]. Finally, we explain whether more angles (e.g., 32 angles) in the rotation set to improve the results even further. Our preliminary experiments using Faster-RCNN with ResNet50 on *COCO-Rot-val* confirmed that the mAP at 32 and 16 angles are the same (mAP=30.0). The mAP will saturate at least around 16 angles.

Comparison with DA and TTA (Sec. 5.2). To demonstrate the superiority of our augmented feature pooling, we describe the performance of DA, TTA, and our proposed method in Sec. 5.2 of our main paper. Table 3 also shows AP_{50} and AP_{75} by our proposed framework, DA, and TTA. We can clearly see that our proposed method has the highest AP_{50} and AP_{75} compared to DA and TTA, as in our main paper.

[3] Note that, in Fig. 2, eight images selected from the original COCO evaluation set, i.e. val2017, are rotated with different rotation angles and combined into a single image. Even though the single image contains different rotation with different rotation angles for each object, our proposed method successfully detects each object.

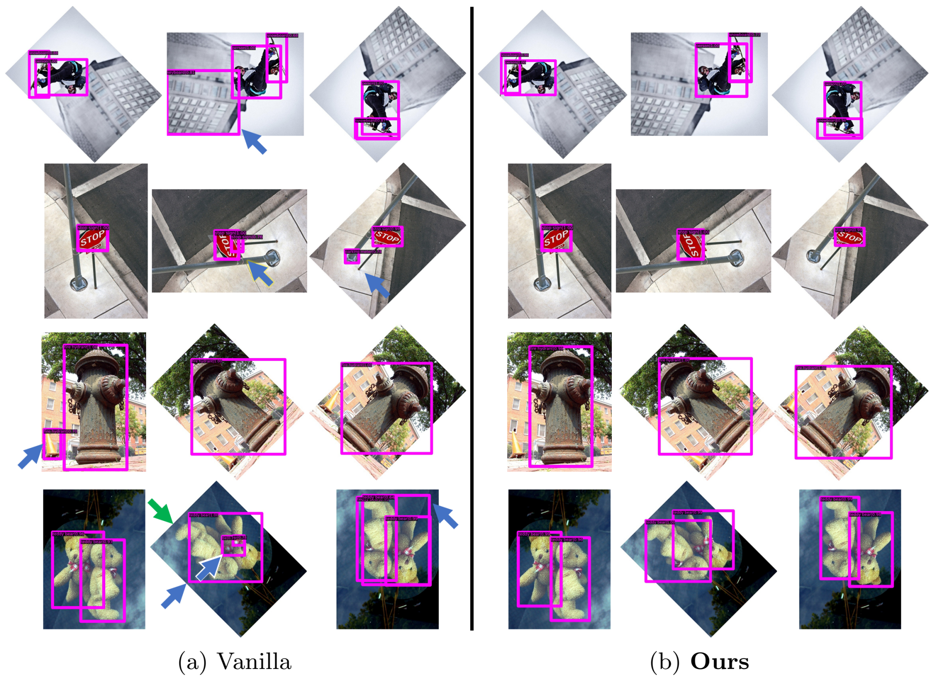


Fig. 1. More example results by our proposed method and vanilla using Faster-RCNN [27] with Swin-Transformer [20]. In the results of vanilla, the green and the blue arrows, i.e. ↖ and ↖, indicate **false negatives** and **false positives**, respectively. Contrary to vanilla backbone feature extraction, our proposed method can detect target objects with accurate bounding boxes for various rotation angles.

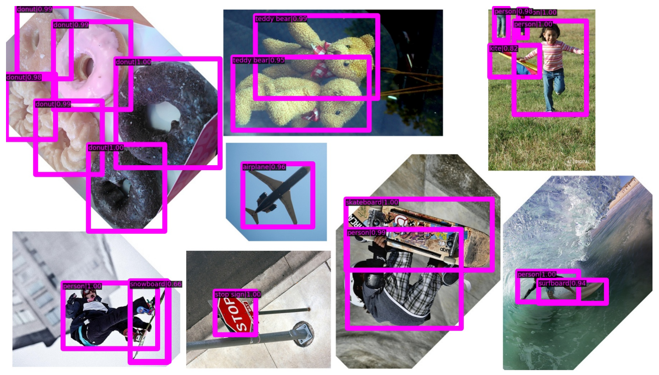


Fig. 2. More example results by our proposed method. Note that eight images selected from the original COCO evaluation set, i.e. val2017, are rotated with different rotation angles and combined into a single image. Even though the single image contains different rotation with different rotation angles for each object, our proposed method successfully detects each object.

Table 1. Performances of AP_{50} and AP_{75} on *COCO-Rot-val* by DA, TTA, and ours. Bold and italic indicate the best and second best results for each column, respectively. MS COCO-train and *COCO-Rot-train* are used as training data, respectively.

(a) Original MS COCO-train. (AP_{50})

Method	AP_{50}
Vanilla	26.2
+ TTA_4	39.7 (+13.5)
+ DA_4	39.3 (+13.1)
+ DA_4+TTA_4	39.5 (+13.3)
Ours$_{16}$	37.3 (+11.1)
+ TTA_4	44.1 (+17.9)
+ DA_4	**45.5 (+19.3)**
+ DA_4 + TTA_4	*45.0 (+18.8)*

(b) *COCO-Rot-train.* (AP_{50})

Method	AP_{50}
Vanilla	41.0
+ TTA_4	44.9 (+3.9)
+ DA_4	41.0 (0.0)
+ DA_4+TTA_4	44.9 (+3.9)
+ *Oracle* DA_4	41.0 (0.0)
+ *Oracle* DA_4+TTA_4	44.5 (+3.5)
Ours$_{16}$	*49.0 (+8.0)*
+ TTA_4	**49.1 (+8.1)**

(c) Original MS COCO-train. (AP_{75})

Method	AP_{75}
Vanilla	16.5
+ TTA_4	26.4 (+9.9)
+ DA_4	21.9 (+5.4)
+ DA_4+TTA_4	25.8 (+9.3)
Ours$_{16}$	21.6 (+5.1)
+ TTA_4	27.6 (+11.1)
+ DA_4	**27.9 (+11.4)**
+ DA_4 + TTA_4	*27.9 (+11.4)*

(d) *COCO-Rot-train.* (AP_{75})

Method	AP_{75}
Vanilla	25.7
+ TTA_4	29.3 (*+3.6*)
+ DA_4	25.9 (+0.2)
+ DA_4+TTA_4	29.3 (*+3.6*)
+ *Oracle* DA_4	25.9 (+0.2)
+ *Oracle* DA_4+TTA_4	*29.7 (+4.0)*
Ours$_{16}$	**31.8 (+6.1)**
+ TTA_4	**31.8 (+6.1)**

Applicability to Modern Object Detection Architectures (Sec. 5.3). We demonstrate the versatility of our proposed method on various object detection networks in Sec. 5.3. Table 2 also shows AP_{50} and AP_{75} of our proposed framework, vanilla with DA_4, our proposed framework with TTA_4 and vanilla with TTA_4, respectively. Note that the performances of mAP are shown in Table 3 in our main paper. Our proposed method substantially improves AP_{50} and AP_{75} for all the detection architectures than DA and TTA.

2 Details of Training Protcols and Datasets

Training Protocols Details and Implementation Details (Sec. 5.1). We describe the details of the training protocols in our experiments. We implemented our code based on MMDetection [4] with PyTorch [23]. We used the default training protocol in MMDetection [4]. The training schedule is 1x[4]. More precisely, we used the SGD optimizer, set the batch size to 16. An initial learning rate, momentum, weight decay are 0.02, 0.9, 10^{-4}, respectively. We trained detectors for 12 epochs with the learning rate decreased by 10x at epoch 8 and 11. We only used random flip for data augmentation. Note that, only for Deformable DETR [35], we used Adam optimizer as in [35], used random flip and auto augmentation [6] for data augmentation, trained detectors for 50 epochs. An initial learning rate and weight decay are 2.0×10^{-4}, $1.0\times^{-4}$, respectively.

[4] In our experiment shown in Table 2 (a) of our main paper, the training schedule is 5x as in [13] for a fair comparison.

Table 2. Overall performance of AP$_{50}$ and AP$_{75}$ on *COCO-Rot-val* by our method and DA and TTA with various object detection networks. Bold and italic indicate the most and next most accurate methods, respectively.

(a) Performance of AP$_{50}$ by ours and DA with various object detection networks.

Baseline	Backbone/Neck	Vanilla	Vanilla+DA$_4$	**Ours$_{16}$**
Faster-RCNN [27]	ResNet50 w/ FPN	41.0	41.0 (0.0)	**49.0 (+8.0)**
RetinaNet [16]	ResNet50 w/ FPN	38.6	38.9 (+0.3)	**46.9 (+8.3)**
FSAF [34]	ResNet50 w/ FPN	38.9	39.3 (+0.4)	**47.8 (+8.9)**
ATSS [32]	ResNet50 w/ FPN	40.6	41.1 (+0.5)	**48.8 (+8.2)**
YOLOF [5]	ResNet50	38.9	39.1 (+0.2)	**45.4 (+6.5)**
D-DETR (++ two-stage) [35]	ResNet50	52.7	54.5 (+1.8)	**58.2 (+5.5)**

(b) Performance of AP$_{75}$ by ours and DA with various object detection networks.

Baseline	Backbone/Neck	Vanilla	Vanilla+DA$_4$	**Ours$_{16}$**
Faster-RCNN [27]	ResNet50 w/ FPN	25.7	25.9 (+0.2)	**31.8 (+6.1)**
RetinaNet [16]	ResNet50 w/ FPN	25.1	25.5 (+0.4)	**31.4 (+6.3)**
FSAF [34]	ResNet50 w/ FPN	25.6	26.1 (+0.5)	**31.5 (+5.9)**
ATSS [32]	ResNet50 w/ FPN	28.2	28.9 (+0.7)	**34.9 (+6.7)**
YOLOF [5]	ResNet50	25.6	25.9 (+0.3)	**29.3 (+3.7)**
D-DETR (++ two-stage) [35]	ResNet50	38.1	39.9 (+1.8)	**42.3 (+4.2)**

(c) Performance of AP$_{50}$ by ours and TTA with various object detection networks.

Baseline	Backbone/Neck	Vanilla	Vanilla+TTA$_4$	**Ours$_{16}$**
Faster-RCNN [27]	ResNet50 w/ FPN	41.0	44.9 (+3.9)	**49.1 (+8.1)**
RetinaNet [16]	ResNet50 w/ FPN	38.6	43.1 (+4.5)	**48.2 (+9.6)**
FSAF [34]	ResNet50 w/ FPN	38.9	43.0 (+4.1)	**48.2 (+9.3)**
ATSS [32]	ResNet50 w/ FPN	40.6	44.1 (+3.5)	**48.7 (+8.1)**
YOLOF [5]	ResNet50	38.9	42.4 (+3.5)	**45.5 (+6.6)**
D-DETR (++ two-stage) [35]	ResNet50	52.7	56.9 (+4.2)	**59.3 (+6.6)**

(d) Performance of AP$_{75}$ by ours and TTA with various object detection networks.

Baseline	Backbone/Neck	Vanilla	Vanilla+TTA$_4$	**Ours$_{16}$**
Faster-RCNN [27]	ResNet50 w/ FPN	25.7	29.3 (+3.6)	**31.8 (+6.1)**
RetinaNet [16]	ResNet50 w/ FPN	25.1	28.7 (+3.6)	**31.1 (+6.0)**
FSAF [34]	ResNet50 w/ FPN	25.6	28.7 (+3.1)	**31.3 (+5.7)**
ATSS [32]	ResNet50 w/ FPN	28.2	31.7 (+3.5)	**34.5 (+6.3)**
YOLOF [5]	ResNet50	25.6	28.6 (+3.0)	**29.6 (+4.0)**
D-DETR (++ two-stage) [35]	ResNet50	38.1	39.6 (+1.5)	**41.3 (+3.2)**

We also provide a pseudo-code of our augmented feature pooling in Algorithm 1. As shown in the pseudo-code, we only add rotation, inverse rotation, and feature pooling to the original forward function of the backbone. It requires only a few lines of changes from the original implementation. Our code including the augmented feature pooling will be available if this paper is accepted.

Datasets Details (Sec. 5.1). As described in Sec. 5.1, we constructed the new datasets, called *COCO-Rot-train* and *COCO-Rot-val*, and evaluated the performance of our augmented feature pooling. We implemented our code for generating the datasets based on PyTorch [23] and MMDetection [4]. To generate a tight bounding box, we randomly rotated the annotated ground truth segmentation mask with the same rotation angle as that of the image, fitted the tight bounding box using this rotated mask. The rotation angle was determined by a

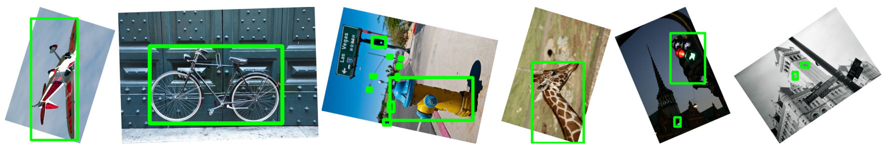
Algorithm 1 Pytorch-like pseudo-code for our augmented feature pooling

```
# forward function in backbone
def forward ($\mathbf{x}^0$, $\Theta$):
    # $x^0$,$\Theta$ :    input image, rotation set
    # $\hat{x}^l$ :        pooled feature map

    $\tilde{X}^l_\Theta = [\ ]$                     # define an empty array
    for $\theta$ in $\Theta = \{\theta_1, \cdots \theta_i\}$:
        $\mathbf{z}^0 = R_\theta(\mathbf{x}^0)$              # rotation
        $\mathbf{z}^l = F^l \circ \cdots \circ F^1 \circ \mathbf{z}^0$    # feature extraction
        $\tilde{\mathbf{x}}^l = R_{-\theta}(\mathbf{z}^l)$            # inverse rotation
        $\tilde{X}^l_\Theta$.append($\tilde{\mathbf{x}}^l$)           # append feature map

    $(\hat{\mathbf{x}}^l)_k = \max_{\theta \in \Theta} (\tilde{X}^l_\Theta)_k$        # feature pooling
    return $\hat{\mathbf{x}}^l$
```

uniform random variable that ranges over all angles, i.e. 0 to 360 degrees. Our *COCO-Rot-train* and *COCO-Rot-val* were generated from the original COCO 2017 training data and the original COCO 2017 validation data, respectively.


(a) Example images with tight bounding boxes of *COCO-Rot-train*

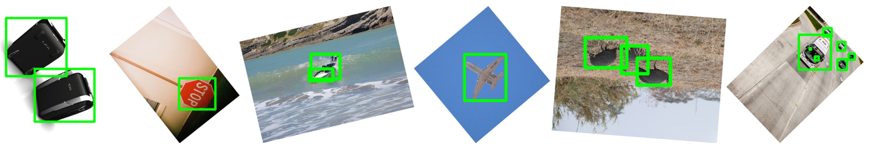

(b) Example images with tight bounding boxes of *COCO-Rot-val*

Fig. 3. Example images with tight bounding boxes of *COCO-Rot*. To generate a tight bounding box, we randomly rotated the annotated ground truth segmentation mask with the same rotation angle as that of the image, fitted the tight bounding box using this rotated mask.

The numbers of images for *COCO-Rot-train* and *COCO-Rot-val* are 118K and 5K, respectively. Figure 3 shows example images with the tight bounding boxes of *COCO-Rot-train* and *COCO-Rot-val*, respectively.

Table 3. Processing speed and FLOP for TTA and ours.

	Flops ↓ $[GFLOPs]$			Proc.speed ↑ $[task/s]$		
	# augmentation: ξ			# augmentation: ξ		
Method	$\xi = 4$	$\xi = 8$	$\xi = 16$	$\xi = 4$	$\xi = 8$	$\xi = 16$
Vanilla		202			61.7	
+ TTA$_\xi$	809	1618	3237	21.2	11.1	5.7
Ours$_\xi$	**459 (x0.57)**	**796 (x0.49)**	**1468 (x0.45)**	**24.5 (x1.15)**	**14.2 (x1.27)**	**7.8 (x1.36)**

3 Computation complexity and time of the proposed method

Our approach can be considered a new variation of TTAs already used in existing networks. Our proposed method can be parallelized, and we think the computational time is reasonable. A comparison of FLOPs (measured at $1280 \times 800 \times 3$ [pix]) and processing speed between the proposed method and baseline (TTA) is shown in Table 1 below. NVIDIA A100 GPUs (four GPUs) were used for our experiments. We used Faster-RCNN with ResNet50. We measured the processing speed (tasks per second) on *COCO-Rot-val*. The proposed method is superior in terms of FLOPs and processing time because our proposed method limits multiple inferences to the backbone as well as detection accuracy. We will add the table in our final version including supplemental material. Further computationally efficient networks are out-of-scope of this paper, while this is an issue to be addressed in the future.

4 Short survey on modern object detection

In this section, we describe a short survey on modern object detection. Note that various comprehensive survey are presented such as [17,36]. As described in our main paper, the architectures of recent object detection consist of three components: backbone, neck, and detection head. Based on the detection head's architecture, the existing object detection algorithms can be classified into single-stage detectors [5,10,16,18,24–26] and two-stage detectors [1,3,12,21,22,27,29,30]. In the single-stage detector framework, the detector predicts class probabilities and bounding box offsets from full images with a single feed-forward CNN. In the two-stage detector framework, the region proposals are generated, and then classifiers are used to determine the category labels of each proposal. In general, single-stage detectors have the advantage of fast and efficient computation, while two-stage detectors have the advantage of high detection accuracy.

While anchors are widely used, anchor-free approaches [15,28,32,34] and keypoint-based approaches [14,33] have been proposed. The center of object as foreground to define positives, and then predicts the distances from positives to the four sides of the object bounding box for detection.

Beyond those CNN-based methods, Transformers have also been employed in detection networks, combining a transformer-based architecture [2,8,35] with a

CNN-based backbone [7,9,11,19,31] or using a transformer-based backbone [20]. The approach of using a transformer in the detection head [2] has the advantage of not requiring post-processing such as NMS.

References

1. Cai, Z., Vasconcelos, N.: Cascade r-cnn: High quality object detection and instance segmentation. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) p. 1–1 (2019). https://doi.org/10.1109/tpami.2019.2956516, `http://dx.doi.org/10.1109/tpami.2019.2956516` 6
2. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: Eur. Conf. Comput. Vis. (ECCV) (2020) 6, 7
3. Chen, K., Pang, J., Wang, J., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Shi, J., Ouyang, W., Loy, C.C., Lin, D.: Hybrid task cascade for instance segmentation. In: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR) (2019) 6
4. Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., Zhang, Z., Cheng, D., Zhu, C., Cheng, T., Zhao, Q., Li, B., Lu, X., Zhu, R., Wu, Y., Dai, J., Wang, J., Shi, J., Ouyang, W., Loy, C.C., Lin, D.: MMDetection: Open mmlab detection toolbox and benchmark. arXiv preprint arXiv:1906.07155 (2019) 3, 4
5. Chen, Q., Wang, Y., Yang, T., Zhang, X., Cheng, J., Sun, J.: You only look one-level feature. In: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR) (2021) 4, 6
6. Cubuk, E.D., Zoph, B., Mane, D., Vasudevan, V., Le, Q.V.: Autoaugment: Learning augmentation policies from data. In: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR) (2019) 3
7. Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., Wei, Y.: Deformable convolutional networks. In: Int. Conf. Comput. Vis. (ICCV). pp. 764–773 (2017) 7
8. Dai, Z., Cai, B., Lin, Y., Chen, J.: Up-detr: Unsupervised pre-training for object detection with transformers. In: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR). pp. 1601–1610 (2021) 6
9. Gao, S., Cheng, M.M., Zhao, K., Zhang, X.Y., Yang, M.H., Torr, P.H.: Res2net: A new multi-scale backbone architecture. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) (2019) 7
10. Ghiasi, G., Lin, T.Y., Le, Q.V.: Nas-fpn: Learning scalable feature pyramid architecture for object detection. In: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR). pp. 7036–7045 (2019) 6
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR) (2016) 7
12. He, K., Gkioxari, G., Dollar, P., Girshick, R.: Mask r-cnn. 2017 IEEE Int. Conf. Comput. Vis. (ICCV) (Oct 2017) 6
13. Kalra, A., Stoppi, G., Brown, B., Agarwal, R., Kadambi, A.: Towards rotation invariance in object detection. In: Int. Conf. Comput. Vis. (ICCV) (2021) 3
14. Law, H., Deng, J.: Cornernet: Detecting objects as paired keypoints. In: Eur. Conf. Comput. Vis. (ECCV). pp. 765–781. Springer Verlag (2018) 6
15. Li, X., Wang, W., Wu, L., Chen, S., Hu, X., Li, J., Tang, J., Yang, J.: Generalized focal loss: Learning qualified and distributed bounding boxes for dense object

detection. Adv. Neural Inform. Process. Syst. (NeurIPS) **33**, 21002–21012 (2020) 6
16. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Int. Conf. Comput. Vis. (ICCV). pp. 2980–2988 (2017) 4, 6
17. Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., Pietikäinen, M.: Deep learning for generic object detection: A survey. Int. J. of Computer Vision (IJCV) **128**(2), 261–318 (2020) 6
18. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: Eur. Conf. Comput. Vis. (ECCV) (2016) 6
19. Liu, Y., Wang, Y., Wang, S., Liang, T., Zhao, Q., Tang, Z., Ling, H.: Cbnet: A novel composite backbone network architecture for object detection. In: Proceedings of the AAAI Conf. on Artificial Intelligence (AAAI). pp. 11653–11660 (2020) 7
20. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. Int. Conf. Comput. Vis. (ICCV) (2021) 2, 7
21. Pang, J., Chen, K., Li, Q., Xu, Z., Feng, H., Shi, J., Ouyang, W., Lin, D.: Towards balanced learning for instance recognition. Int. J. Comput. Vis. (IJCV) **129**(5), 1376–1393 (2021) 6
22. Pang, J., Chen, K., Shi, J., Feng, H., Ouyang, W., Lin, D.: Libra r-cnn: Towards balanced learning for object detection. In: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR) (2019) 6
23. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Adv. Neural Inform. Process. Syst. (NeurIPS) **32**, 8026–8037 (2019) 3, 4
24. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR). pp. 779–788 (2016) 6
25. Redmon, J., Farhadi, A.: Yolo9000: better, faster, stronger. In: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR). pp. 7263–7271 (2017) 6
26. Redmon, J., Farhadi, A.: Yolov3: An incremental improvement (2018) 6
27. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Adv. Neural Inform. Process. Syst. (NeurIPS) (2015) 2, 4, 6
28. Tian, Z., Shen, C., Chen, H., He, T.: Fcos: Fully convolutional one-stage object detection. arXiv preprint arXiv:1904.01355 (2019) 6
29. Vu, T., Jang, H., Pham, T.X., Yoo, C.D.: Cascade rpn: Delving into high-quality region proposal network with adaptive convolution. In: Adv. Neural Inform. Process. Syst. (NeurIPS) (2019) 6
30. Wu, Y., Chen, Y., Yuan, L., Liu, Z., Wang, L., Li, H., Fu, Y.: Rethinking classification and localization for object detection. arXiv (2019) 6
31. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR). pp. 1492–1500 (2017) 7
32. Zhang, S., Chi, C., Yao, Y., Lei, Z., Li, S.Z.: Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. arXiv preprint arXiv:1912.02424 (2019) 4, 6
33. Zhou, X., Wang, D., Krähenbühl, P.: Objects as points. In: arXiv preprint arXiv:1904.07850 (2019) 6

34. Zhu, C., He, Y., Savvides, M.: Feature selective anchor-free module for single-shot object detection. In: IEEE Conf. Comput. Vis. Pattern Recog. (CVPR). pp. 840–849 (2019) 4, 6
35. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable detr: Deformable transformers for end-to-end object detection. In: Int. Conf. on Learn. Represent. (2021) 3, 4, 6
36. Zou, Z., Shi, Z., Guo, Y., Ye, J.: Object detection in 20 years: A survey. arXiv preprint arXiv:1905.05055 (2019) 6