Supplementary for "EAutoDet: Efficient Architecture Search for Object Detection"

Xiaoxing Wang[1], Jiale Lin[1], Juanping Zhao[2], Xiaokang Yang[1], and Junchi Yan[1](✉)

[1] Department of CSE & MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University

[2] Guangdong OPPO Mobile Telecommunications Co., Ltd.

{figure1_wxx linjiale, xkyang, yanjunchi}@sjtu.edu.cn

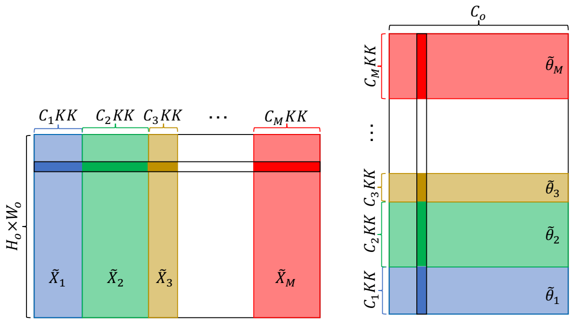
zhaojuanping1325@oppo.com

1 Rationality of Transformation for Concatenation Layers

To explain the rationality of transformation for concatenation layers, we can prove the following proposition.

Proposition 1 *The output of a concatenation layer followed by a convolution layer is equivalent to the sum of separate convolutions on the inputs.*

Suppose we have M features $\boldsymbol{X}_m \in \mathbb{R}^{N\times C_m\times H\times W}, m \in [1, M]$. By concatenating theses features along channel dimension, we can obtain a new feature $\boldsymbol{X} \in \mathbb{R}^{N\times(\sum_{m=1}^{M} C_m)\times H\times W}$. A convolution with weights $\boldsymbol{\theta} \in \mathbb{R}^{C_o\times(\sum_{m=1}^{M} C_m)\times K\times K}$ is then applied on the concatenated feature $\boldsymbol{X}$. Im2col operation transfers convolution to matrix multiplication, and we get the input and weight matrix $\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{\theta}}$, as shown in Fig. 1. $\tilde{\boldsymbol{X}} \cdot \tilde{\boldsymbol{\theta}} = \sum_{m=1}^{M} \tilde{\boldsymbol{X}}_m \cdot \tilde{\boldsymbol{\theta}}_m$, which is the sum of M multiplications of small matrix, where $\tilde{\boldsymbol{X}}_m \cdot \tilde{\boldsymbol{\theta}}_m$ is exactly the convolution on input feature $\boldsymbol{X}_m$. Consequently, Proposition 1 is proved.



(a) Input matrix (b) Weight matrix

Fig. 1. Illustration of an input and weight matrix $\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{\theta}}$.

Table 1. Search space size: **s**mall, **m**edium, **l**arge, e**x**tra large. Total size equals the multiplication of the backbone and FPN space sizes. Four supernets in various depth/width are designed. Details of the four search spaces are in the supplementary.

Supernet	**Size of Search Space**		
	Backbone	**FPN**	**Total**
EAutoDet-s	7.9×10^{11}	9.8×10^{24}	7.7×10^{36}
EAutoDet-m	3.8×10^{18}	5.2×10^{30}	2.0×10^{49}
EAutoDet-l	1.8×10^{25}	2.8×10^{36}	5.0×10^{61}
EAutoDet-x	8.7×10^{31}	1.5×10^{42}	1.3×10^{74}

2 Details of our Search Spaces

The size of search space is given in Table 1

Marco Architectures. To abosrb the knowledge of architectures of YOLO models that are well designed by experts, we refer to YOLOv5 [9] and build four supernets with various depths and widths, denoted as s (small), m (medium), l (large), and x (extra large). The depth and width of various types of supernet are illustrated in Table 2, where 'C' is the number of base output channels indicating the supernet width, and 'M' is the number of bottleneck cells in the C3 block, affecting the supernet depth. The details of our search space for the backbone and FPN modules are introduced as follows.

Backbone Search Space. 1) For the down-sampling operator, we design four candidates: {1x1 convolution, 3x3 convolution, 5x5 convolution, 3x3 dilated convolution} and three candidate expansion rates for output channel: {0.5, 0.75, 1.0}; 2) For the bottleneck cell, which consists of two convolutions, we search output channels for both convolutions with candidates {0.5, 0.75, 1.0}, and kernel settings for the second convolution with candidates {3x3 convolution, 5x5 convolution, 3x3 dilated convolution}; 3) For the C3-block, which consists of M bottleneck cells, we search for the expansion rate for its output channels among two candidates: {0.75, 1.0}. We adopt macro search space in this work, where architectures for bottlenecks and C3-blocks from different layers are independently searched. Suppose a backbone contains L_D down-sampling layers, L_C C3-blocks and L_B Bottlenecks, the size of search space is $(4 \times 3)^{L_D} \cdot 2^{L_C} \cdot (3 \times 3)^{L_B}$.

FPN Search Space. This work extracts three scales of features and builds supernets for both Top-down and Bottom-up fusion blocks. Each node in the supernet indicates a feature map connecting with all its predecessors, and each edge owns four candidate operations: {1x1 convolution, 3x3 convolution, 5x5 convolution, 3x3 dilated convolution} with three possible expansion rates for output channel: {0.5,0.75,1.0}. To derive the final architecture, each node in top-down and bottom-up fusion blocks preserves top-2 connections, and each connection will only preserve the best operation. Since we have three feature scales, there are 6 connections in each fusion block, leading to $(4 \times 3)^6 \times \left[\binom{3}{2} \times \binom{4}{2} \times \binom{5}{2}\right] \approx 5.4 \times 10^8$ candidate connection types for each feature fusion block in total. Furthermore, we concatenate a C3 block with K_B Bottlenecks after each feature

Table 2. Structures of the four supernet built based on YOLOv5. 'C' is the number of output channels (width), and 'M' denotes the number of bottleneck cells in the C3 block (depth). The last line indicates the statistics of the supernet structures used to calculate the size of the search space. Specifically, L_D, L_C, L_B is the number of down-sampling layers, C3-blocks and Bottlenecks in the backbone, and K_B is the number of Bottlenecks in each fusion block in the FPN module.

Module	Block	AutoYOLO-s	AutoYOLO-m	AutoYOLO-l	AutoYOLO-x
Backbone	Focus	C=32	C=48	C=64	C=80
	Down-sample	C=64	C=96	C=128	C=160
	C3	C=64, M=1	C=96, M=2	C=128, M=3	C=160, M=4
	Down-sample	C=128	C=192	C=256	C=320
	C3	C=128, M=3	C=192, M=6	C=256, M=9	C=320, M=12
	Down-sample	C=256	C=384	C=512	C=640
	C3	C=256, M=3	C=384, M=6	C=512, M=9	C=640, M=12
	Down-sample	C=512	C=768	C=1024	C=1280
	SPP	C=512	C=768	C=1024	C=1280
Top-down (FPN)	Feature Fusion	$\begin{bmatrix} 32/\ C{=}512 \\ 16/\ C{=}256 \\ 8/\ C{=}128 \end{bmatrix}$	$\begin{bmatrix} 32/\ C{=}768 \\ 16/\ C{=}384 \\ 8/\ C{=}192 \end{bmatrix}$	$\begin{bmatrix} 32/\ C{=}1024 \\ 16/\ C{=}512 \\ 8/\ C{=}256 \end{bmatrix}$	$\begin{bmatrix} 32/\ C{=}1280 \\ 16/\ C{=}640 \\ 8/\ C{=}320 \end{bmatrix}$
	C3	$\begin{bmatrix} 32/\ C{=}512,\ M{=}1 \\ 16/\ C{=}256,\ M{=}1 \\ 8/\ C{=}128,\ M{=}1 \end{bmatrix}$	$\begin{bmatrix} 32/\ C{=}768,\ M{=}2 \\ 16/\ C{=}384,\ M{=}2 \\ 8/\ C{=}192,\ M{=}2 \end{bmatrix}$	$\begin{bmatrix} 32/\ C{=}1024,\ M{=}3 \\ 16/\ C{=}512,\ M{=}3 \\ 8/\ C{=}256,\ M{=}3 \end{bmatrix}$	$\begin{bmatrix} 32/\ C{=}1280,\ M{=}4 \\ 16/\ C{=}640,\ M{=}4 \\ 8/\ C{=}320,\ M{=}4 \end{bmatrix}$
Bottom-up (FPN)	Feature Fusion	$\begin{bmatrix} 32/\ C{=}512 \\ 16/\ C{=}256 \\ 8/\ C{=}128 \end{bmatrix}$	$\begin{bmatrix} 32/\ C{=}768 \\ 16/\ C{=}384 \\ 8/\ C{=}192 \end{bmatrix}$	$\begin{bmatrix} 32/\ C{=}1024 \\ 16/\ C{=}512 \\ 8/\ C{=}256 \end{bmatrix}$	$\begin{bmatrix} 32/\ C{=}1280 \\ 16/\ C{=}640 \\ 8/\ C{=}320 \end{bmatrix}$
	C3	$\begin{bmatrix} 32/\ C{=}512,\ M{=}1 \\ 16/\ C{=}256,\ M{=}1 \\ 8/\ C{=}128,\ M{=}1 \end{bmatrix}$	$\begin{bmatrix} 32/\ C{=}768,\ M{=}2 \\ 16/\ C{=}384,\ M{=}2 \\ 8/\ C{=}192,\ M{=}2 \end{bmatrix}$	$\begin{bmatrix} 32/\ C{=}1024,\ M{=}3 \\ 16/\ C{=}512,\ M{=}3 \\ 8/\ C{=}256,\ M{=}3 \end{bmatrix}$	$\begin{bmatrix} 32/\ C{=}1280,\ M{=}4 \\ 16/\ C{=}640,\ M{=}4 \\ 8/\ C{=}320,\ M{=}4 \end{bmatrix}$
Statistics		L_D=4,L_C=3,L_B=7,K_B=3	L_D=4,L_C=3,L_B=14,K_B=6	L_D=4,L_C=3,L_B=21,K_B=9	L_D=4,L_C=3,L_B=28,K_B=12

fusion block, which has $2 \cdot (3 \times 3)^{K_B}$. Since we have two fusion blocks: Top-down and Bottom-up block, the size of search space for FPN is $\{(4 \times 3)^6 \times \left[\binom{3}{2} \times \binom{4}{2} \times \binom{5}{2}\right] \times 2 \cdot (3 \times 3)^{K_B}\}^2$.

3 Details of Experimental Settings

We search on MS-COCO 2017 detection dataset [14] and evaluation on the test set of MS-COCO and DOTA-v1.0 benchmark. All our models are trained from scratch without pre-training on ImageNet.

Search Settings. We construct a supernet and define architecture parameters to represent the importance of candidate operations and connections. Unlike DARTS that shares the same cell structure, we independently search architectures for each C3 block and Bottleneck. The training set of MS-COCO is divided into two parts for training architecture parameters and network weights, respectively. The final architecture is derived after alternately optimizing architecture parameters and network weights for 50 epochs by an SGD optimizer.

Evaluation Settings. The discovered architectures are trained from scratch for 300 epochs by an SGD optimizer. We directly utilize the hyper-parameters provided by YOLOv5 for a fair comparison. Our experiments are conducted on V100 GPU. To fairly compare the speed (FPS) with YOLO methods, we convert the trained models to the style of YOLOv4 [1] and evaluate the FPS

Table 3. Detailed comparison between the discovered architectures and original YOLOv5 models. MAP is tested on the test set of MS-COCO.

Model	**#Params (M)**	**FLOPs (G)**	**FPS**	**mAP(%)**
YOLOv5-s	7.3	17.1	113	36.9
YOLOv5-m	21.4	51.4	88	43.9
YOLOv5-l	47.1	112.5	59	46.8
YOLOv5-x	87.8	219.0	43	49.1
AutoYOLO-s	9.1	24.9	120	40.1
AutoYOLO-m	28.1	60.8	70	45.2
AutoYOLO-l	34.4	115.4	59	47.9
AutoYOLO-x	86.0	225.3	41	49.2

on the Darknet platform [19], which is written in C and CUDA. Besides, we evaluate the generalization of the discovered architectures by transferring them to rotation detection task. Specifically, we train models on the training set of DOTA-v1.0 from scratch for 300 epochs and evaluate on the validation and test sets. Notice that we train and test on a single input scale unlike previous works [6, 28] that adopt multi-scale training technique and random rotation augmentation. All our experiments are trained and tested on the V100 GPU, and our models are trained on PyTorch platform.

4 Comparison to YOLOv5 models.

Table 3 shows the detailed information of our models and YOLOv5 series models, including number of parameters, FLOPs, and mAP on the test set of MS-COCO. We observe that our models archieve better performance than YOLOv5 with similar parameters and FPS, showing the effectiveness of our search method. We will open-source our search and evaluation codes. Table 4 compares with prior works on the test set of MS-COCO.

5 Results on Rotation Detection Benchmark: DOTA

We utilize Circular Smooth Label (CSL) technique [27] to obtain robust angular prediction through classification without suffering boundary conditions. The baseline adopts RetinaNet [13] detection framework with ResNet152 [7] backbone and FPN [12] module. Our models are trained with rotation classification loss used in CSL [27] from scratch for 300 epochs. Besides, we compare to YOLOv5 based on the open-sourced codes [10]. Results are shown in Table 5. We observe that: **1)** Our EAutoDet-s achieves competitive performance over many prior works in ResNet backbone; **2)** EAutoDet-m outperforms the baseline (CSL [27] with ResNet-152 backbone) by 0.8% mAP_{50}; **3)** EAutoDet-s model surpasses original YOLOv5-s model by more than 1.7% mAP_{50} and EAutoDet-m surpasses YOLOv5-m by over 0.7%.

Table 4. Comparison with prior works on the COCO test-dev. FPS for YOLOv5 and our method are calculated on a single V100 GPU, and results for other methods are directly obtained from their papers. Different blocks indicate models with various inference speeds and prediction performance. '†': The results are obtained by our experiments. '-': The value is not provided by the original paper. '*': The unit of search cost is TPU-days, while the unit of other methods is GPU-days. '‡': SPNet[8] shows the search cost on VOC is 26 GPU-days, and is six times lower than that on COCO.

Method	**Resolution**	**FPS**	**#Params (M)**	**mAP (%)**	**AP$_{50}$ (%)**	**AP$_{75}$ (%)**	**AP$_S$ (%)**	**AP$_M$ (%)**	**AP$_L$ (%)**	**Search Cost**
YOLOv4 [1]	416	96	-	41.2	62.8	44.3	20.4	44.4	56.0	-
YOLOv5s† [9]	640	113	7.3	36.9	56.0	40.0	19.9	41.1	46.0	-
EfficientDet-D0 [20]	512	98	3.9	33.8	52.2	35.8	12.0	38.3	51.2	-
NAS-FPN [4]	640	24	60.3	39.9	-	-	-	-	-	333*
NAS-FCOS@128 [23]	1333×800	-	27.8	37.9	-	-	-	-	-	28
SpineNet-49S [3]	640	-	11.9	39.5	59.3	43.1	20.9	42.2	54.3	-
SM-NAS:E2 [29]	800×600	25	-	40.0	58.2	43.4	21.1	42.4	51.7	187
EAutoDet-s (ours)	**640**	**120**	**9.1**	**40.1**	**58.7**	**43.5**	**21.7**	**43.8**	**50.5**	**1.4**
YOLOv3 + ASFF [15]	416	54	-	40.6	60.6	45.1	20.3	44.2	54.1	-
YOLOv4 [1]	512	83	-	43.0	64.9	46.5	24.3	46.1	55.2	-
YOLOv4-csp [21]	512	80†	43	46.2	64.8	50.2	24.6	50.4	61.9	-
YOLOv5m† [9]	640	88	21.4	43.9	62.5	47.6	25.1	48.1	54.9	-
EfficientDet-D1 [20]	640	74	6.6	39.6	58.6	42.3	17.9	44.3	56.0	-
DetNAS [2]	1333×800	-	-	42.0	63.9	45.8	24.9	45.1	56.8	44
NAS-FPN [4]	1024	13	60.3	44.2	-	-	-	-	-	333*
Auto-FPN [25]	800	-	32.6	40.5	61.5	43.8	25.6	44.9	51.0	16
NAS-FCOS@256 [23]	1333×800	-	57.3	43.0	-	-	-	-	-	28
SpineNet-49 [3]	640	-	28.5	42.8	62.3	46.1	23.7	45.2	57.3	-
SM-NAS:E3 [29]	800×600	20	-	42.8	61.2	46.5	23.5	45.5	55.6	187
Hit-Detector [5]	1200×800	-	27.1	41.4	62.4	45.9	25.2	45.0	54.1	-
OPA-FPN@64 [11]	1333×800	22	29.5	41.9	-	-	-	-	-	4
EAutoDet-m (ours)	**640**	**70**	**28.1**	**45.2**	**63.5**	**49.1**	**25.7**	**49.1**	**57.3**	**2.2**
YOLOv3 + ASFF [15]	608	46	-	42.4	63.0	47.4	25.5	45.7	52.3	-
YOLOv4 [1]	608	62	-	43.5	65.7	47.3	26.7	46.7	53.3	-
YOLOv4-csp [21]	640	65†	53	47.5	66.2	51.7	28.2	51.2	59.8	-
EAutoDet-csp (ours)	**640**	**55**	**49.8**	**47.8**	**66.1**	**51.9**	**28.6**	**51.5**	**60.1**	**4.2**
YOLOv5l† [9]	640	59	47.1	46.8	65.4	50.9	27.7	51.0	58.5	-
EfficientDet-D2 [20]	768	57	8.1	43.0	62.3	46.2	22.5	47.0	58.4	-
SPNet(BNB) [8]	1333×800	10	-	45.6	64.3	49.6	28.4	48.4	60.1	156‡
SM-NAS:E5 [29]	1333×800	9	-	45.9	64.6	48.6	27.1	49.0	58.0	187
OPA-FPN@160 [11]	1333×800	13	60.6	47.0	-	-	-	-	-	4
EAutoDet-l (ours)	**640**	**59**	**34.4**	**47.9**	**66.3**	**52.0**	**28.3**	**52.0**	**59.9**	**4.5**
YOLOv3 + ASFF [15]	800	29	-	43.9	64.1	49.2	27.0	46.6	53.4	-
YOLOv5x† [9]	640	43	87.8	49.1	67.5	53.6	30.2	53.4	61.4	-
EfficientDet-D3 [20]	896	35	12	45.8	65.0	49.3	26.6	49.4	59.8	-
SPNet(XB) [8]	1333×800	6	-	47.4	65.7	51.9	29.6	51.0	60.4	156‡
EAutoDet-x (ours)	**640**	**41**	**86.0**	**49.2**	**67.5**	**53.6**	**30.4**	**53.4**	**61.5**	**22**

References

1. Bochkovskiy, A., Wang, C.Y., Liao, H.Y.M.: Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934 (2020)

Table 5. Comparison on the test set of oriented bounding boxes (OBB) task in DOTA-v1.0 benchmark. R152 denotes ResNet-152 [7] and H-104 denotes Hourglass-104 [17]. †: Models are trained with multi-scale training techniques, while others are not. *: two-stage detection framework, while others belong to one-stage framework.

Method	**PL**	**BD**	**BR**	**GTF**	**SV**	**LV**	**SH**	**TC**	**BC**	**ST**	**SBF**	**RA**	**HA**	**SP**	**HC**	**mAP$_{50}$**
ReDet(ReR50) [6]*	88.79	82.64	53.97	74.00	78.13	84.06	88.04	**90.89**	87.78	85.75	61.76	60.39	75.96	68.07	63.39	76.25
CenterMap(R101) [22]*	89.83	84.41	**54.60**	70.25	77.66	78.32	87.19	90.66	84.89	85.27	56.46	69.23	74.13	71.56	66.06	76.03
CSL(R152) [27]$^{\star\ \dagger}$	**90.13**	84.43	54.57	68.13	77.32	72.98	85.94	90.74	85.95	86.36	63.42	65.82	74.06	73.67	70.08	76.24
O^2-DNet(H104) [24]	89.31	82.14	47.33	61.21	71.32	74.03	78.62	90.76	82.23	81.36	60.93	60.17	58.21	66.98	61.03	71.04
DAL(R101)[16]	88.61	79.69	46.27	70.37	65.89	76.10	78.53	90.84	79.98	78.41	58.71	62.02	69.23	71.32	60.65	71.78
P-RSDet(R101) [32]	88.58	77.83	50.44	69.29	71.10	75.79	78.66	90.88	80.10	81.71	57.92	63.03	66.30	69.77	63.13	72.30
BBAVectors(R101) [30]	88.35	79.96	50.69	62.18	78.43	78.98	87.94	90.85	83.58	84.35	54.13	60.24	65.22	64.28	55.70	72.32
DRN(H104) [18]†	89.71	82.34	47.22	64.10	76.22	74.43	85.84	90.57	86.18	84.89	57.65	61.93	69.30	69.63	58.48	73.23
DCL(R152) [26]†	89.10	84.13	50.15	73.57	71.48	58.13	78.00	**90.89**	86.64	86.78	67.97	67.25	65.63	74.06	67.05	74.06
PolarDet(R101) [31]†	89.65	**87.07**	48.14	70.97	78.53	80.34	87.45	90.76	85.63	86.87	61.64	**70.32**	71.92	73.09	67.15	76.64
GWD(R152) [28]†	86.96	83.88	54.36	**77.53**	74.41	68.48	80.34	86.62	83.41	85.55	**73.47**	67.77	72.57	75.76	73.40	76.30
CSL(YOLOv5-s) [10]	88.93	76.07	50.97	59.61	81.38	84.00	88.19	90.81	80.52	87.63	47.38	62.85	68.27	80.63	62.74	74.00
CSL(**EAutoDet-s**)	88.64	84.78	51.02	62.04	81.30	84.02	88.23	90.82	85.79	87.36	52.20	65.68	73.72	74.33	65.93	75.72
CSL(YOLOv5-m) [10]	88.36	85.24	53.86	62.35	81.19	84.72	**88.57**	90.77	80.98	**88.09**	53.92	61.90	**75.99**	80.43	**68.59**	76.33
CSL(**EAutoDet-m**)	89.04	86.61	53.83	63.05	**81.51**	**85.12**	88.46	90.77	**88.21**	87.93	56.04	60.96	75.96	**82.13**	66.18	**77.05**

2. Chen, Y., Yang, T., Zhang, X., Meng, G., Xiao, X., Sun, J.: Detnas: Backbone search for object detection. NeurIPS (2019)
3. Du, X., Lin, T.Y., Jin, P., Ghiasi, G., Tan, M., Cui, Y., Le, Q.V., Song, X.: Spinenet: Learning scale-permuted backbone for recognition and localization. In: CVPR (2020)
4. Ghiasi, G., Lin, T.Y., Le, Q.V.: Nas-fpn: Learning scalable feature pyramid architecture for object detection. In: CVPR (2019)
5. Guo, J., Han, K., Wang, Y., Zhang, C., Yang, Z., Wu, H., Chen, X., Xu, C.: Hit-detector: Hierarchical trinity architecture search for object detection. In: CVPR (2020)
6. Han, J., Ding, J., Xue, N., Xia, G.: Redet: A rotation-equivariant detector for aerial object detection. In: CVPR (2021)
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
8. Jiang, C., Xu, H., Zhang, W., Liang, X., Li, Z.: Sp-nas: Serial-to-parallel backbone search for object detection. In: CVPR (2020)
9. Jocher, G.: Yolov5 documentation. https://docs.ultralytics.com/ (May 2020)
10. Kaixuan, H.: Yolov5 for oriented object detection. https://github.com/hukaixuan19970627/yolov5_obb (2020)
11. Liang, T., Wang, Y., Tang, Z., Hu, G., Ling, H.: Opanas: One-shot path aggregation network architecture search for object detection. In: CVPR (2021)
12. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: CVPR (2017)
13. Lin, T., Goyal, P., Girshick, R.B., He, K., Dollár, P.: Focal loss for dense object detection. TPAMI **42**(2), 318–327 (2020)
14. Lin, T., Maire, M., Belongie, S.J., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: common objects in context. In: Fleet, D.J., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV (2014)
15. Liu, S., Huang, D., Wang, Y.: Learning spatial fusion for single-shot object detection. arXiv preprint arXiv:1911.09516 (2019)

16. Ming, Q., Zhou, Z., Miao, L., Zhang, H., Li, L.: Dynamic anchor learning for arbitrary-oriented object detection. In: Proceedings of the AAAI Conference on Artificial Intelligence (2021)
17. Newell, A., Yang, K., Deng, J.: Stacked hourglass networks for human pose estimation. In: ECCV (2016)
18. Pan, X., Ren, Y., Sheng, K., Dong, W., Yuan, H., Guo, X., Ma, C., Xu, C.: Dynamic refinement network for oriented and densely packed object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 11207–11216 (2020)
19. Redmon, J.: Darknet: Open source neural networks in c. http://pjreddie.com/darknet/ (2013–2016)
20. Tan, M., Pang, R., Le, Q.V.: Efficientdet: Scalable and efficient object detection. In: CVPR (2020)
21. Wang, C.Y., Bochkovskiy, A., Liao, H.Y.M.: Scaled-yolov4: Scaling cross stage partial network. In: CVPR (2021)
22. Wang, J., Yang, W., Li, H., Zhang, H., Xia, G.: Learning center probability map for detecting objects in aerial images. IEEE Trans. Geosci. Remote. Sens. (2021)
23. Wang, N., Gao, Y., Chen, H., Wang, P., Tian, Z., Shen, C., Zhang, Y.: Nas-fcos: Fast neural architecture search for object detection. In: CVPR (2020)
24. Wei, H., Zhang, Y., Chang, Z., Li, H., Wang, H., Sun, X.: Oriented objects as pairs of middle lines. ISPRS Journal of Photogrammetry and Remote Sensing **169**, 268–279 (2020)
25. Xu, H., Yao, L., Li, Z., Liang, X., Zhang, W.: Auto-fpn: Automatic network architecture adaptation for object detection beyond classification. In: ICCV (2019)
26. Yang, X., Hou, L., Zhou, Y., Wang, W., Yan, J.: Dense label encoding for boundary discontinuity free rotation detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 15819–15829 (2021)
27. Yang, X., Yan, J.: Arbitrary-oriented object detection with circular smooth label. In: ECCV (2020)
28. Yang, X., Yan, J., Qi, M., Wang, W., Xiaopeng, Z., Qi, T.: Rethinking rotated object detection with gaussian wasserstein distance loss. In: International Conference on Machine Learning (2021)
29. Yao, L., Xu, H., Zhang, W., Liang, X., Li, Z.: Sm-nas: structural-to-modular neural architecture search for object detection. In: AAAI (2020)
30. Yi, J., Wu, P., Liu, B., Huang, Q., Qu, H., Metaxas, D.: Oriented object detection in aerial images with box boundary-aware vectors. In: Proceedings of the IEEE Winter Conference on Applications of Computer Vision. pp. 2150–2159 (2021)
31. Zhao, P., Qu, Z., Bu, Y., Tan, W., Guan, Q.: Polardet: A fast, more precise detector for rotated target in aerial images. International Journal of Remote Sensing **42**(15), 5821–5851 (2021)
32. Zhou, L., Wei, H., Li, H., Zhao, W., Zhang, Y., Zhang, Y.: Arbitrary-oriented object detection in remote sensing images based on polar coordinates. IEEE Access **8**, 223373–223384 (2020)