# Adaptive Rotated Convolution for Rotated Object Detection

Yifan Pu[1*] Yiru Wang[2*] Zhuofan Xia[1] Yizeng Han[1] Yulin Wang[1]
Weihao Gan[3] Zidong Wang[1] Shiji Song[1] Gao Huang[1,4✉]
[1]Department of Automation, BNRist, Tsinghua University [2]SenseTime Research
[3]Mashang Consumer Finance Co., Ltd. [4]Beijing Academy of Artificial Intelligence
pyf20@mails.tsinghua.edu.cn, wangyiru@sensetime.com, gaohuang@tsinghua.edu.cn

## Abstract

*Rotated object detection aims to identify and locate objects in images with arbitrary orientation. In this scenario, the oriented directions of objects vary considerably across different images, while multiple orientations of objects exist within an image. This intrinsic characteristic makes it challenging for standard backbone networks to extract high-quality features of these arbitrarily orientated objects. In this paper, we present Adaptive Rotated Convolution (ARC) module to handle the aforementioned challenges. In our ARC module, the convolution kernels rotate adaptively to extract object features with varying orientations in different images, and an efficient conditional computation mechanism is introduced to accommodate the large orientation variations of objects within an image. The two designs work seamlessly in rotated object detection problem. Moreover, ARC can conveniently serve as a plug-and-play module in various vision backbones to boost their representation ability to detect oriented objects accurately. Experiments on commonly used benchmarks (DOTA and HRSC2016) demonstrate that equipped with our proposed ARC module in the backbone network, the performance of multiple popular oriented object detectors is significantly improved* (e.g. *+3.03% mAP on Rotated RetinaNet and +4.16% on CFA). Combined with the highly competitive method Oriented R-CNN, the proposed approach achieves state-of-the-art performance on the DOTA dataset with 81.77% mAP. Code is available at* https://github.com/LeapLabTHU/ARC.

## 1. Introduction

Rotated object detection has become an emerging research topic in recent years [74, 89, 56]. Different from generic object detection, where object instances are assumed to be aligned with the image axes, objects in the natural scenes are prone to be placed with arbitrary orientation. This phenomenon is commonly observed in the field of scene text detection [93, 41, 40], face detection [38, 84], and aerial image recognition[32, 49, 74], *etc*. In particular, in Embodied AI [11, 77, 50, 80, 13] tasks, as the agents need to explore and interact with the environment [4, 3, 42], the captured images may contain an object from arbitrary viewpoints [15, 59]. This poses great challenges for detection algorithms to accurately locate oriented objects.

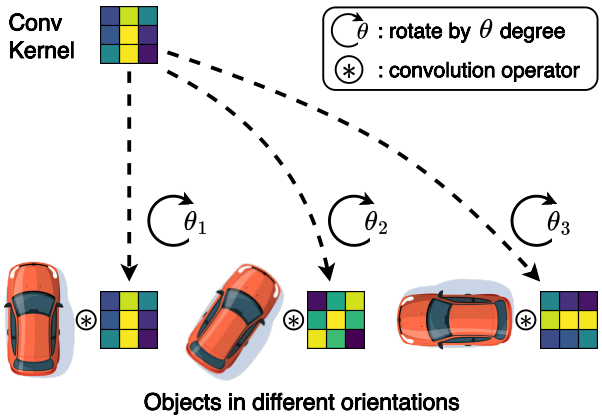



Figure 1. **The motivation of our work**. In the rotated object detection scenario, object instances with similar visual appearance are placed with arbitrary orientation (*e.g.*, the cars). As a result, it is reasonable to rotate the convolution kernels according to the orientation of the objects in a data-dependent manner rather than processing the image samples with the same static kernel.

Recently, considerable progress has been achieved in detecting rotated objects. For instance, various rotated object representations [79, 17, 44, 34, 91] and more suitable loss functions for these object representations [56, 6, 88, 90, 92] have been extensively analyzed. The mechanisms of the rotated region of interest extraction [9, 76] and label assignment strategies [52, 53] have also been well explored. Moreover, the structure of the detection networks, including the neck [89, 86, 87] and head [19, 33] of detectors, and rotated region proposal networks [76, 7] has been comprehensively studied as well. However, little effort has been made in the design of a proper *backbone* feature extractor.

*Equal Contribution. ✉ Corresponding author.

The quality of the features extracted by backbone networks is crucial to many vision tasks. In particular, rotated object detection raises great challenges for backbone network design, since the orientation of objects varies *across different images*. In the meanwhile, multiple orientations of objects also exist *within an image*. Despite the significant differences between the images with generic items and those with oriented objects, the design of conventional visual backbones has mostly ignored the inherent characteristics. Therefore, the architecture of standard backbone models may be sub-optimal in the rotated object detection task.

In this paper, we address the above challenges by proposing a simple yet effective **Adaptively Rotated Convolution (ARC)** module. In this module, the convolution kernel *adaptively rotates* to adjust the parameter conditioned on each input (Fig. 1), where the rotation angle is predicted by a routing function in a *data-dependent* manner. Furthermore, an efficient conditional computation technique is employed, which endows the detector with more adaptability to handle objects with various orientations within an image. Specifically, multiple kernels are rotated individually and then combined together before being applied for convolution operations. This combine-and-compute procedure is equivalent to performing convolution with different kernels separately and then summing up the obtained results, yet the computation could be significantly reduced [22]. The two designs work seamlessly, effectively enlarging the parameter space and elegantly endowing the network with more flexibility to detect objects in different orientations.

The proposed ARC module can conveniently serve as a plug-and-play module in convolution layers with arbitrary kernel size. As a result, any backbone network with convolution layers can enjoy the powerful representation ability of rotated objects by using the ARC module. For example, we can replace the convolution layers with the ARC module in the commonly used backbone network ResNet [31] to build the proposed backbone network ARC-ResNet.

We evaluate our method on two popular rotated object detection benchmarks (DOTA [74] and HRSC2016 [49]). Extensive experiments validate that with the backbone network equipped with our proposed adaptive rotated convolution (ARC) module, the performance of various popular oriented object detectors can be effectively enhanced on both datasets. When combined with a highly competitive method Oriented R-CNN [76], our approach achieves state-of-the-art performance on the DOTA-v1.0 benchmark.

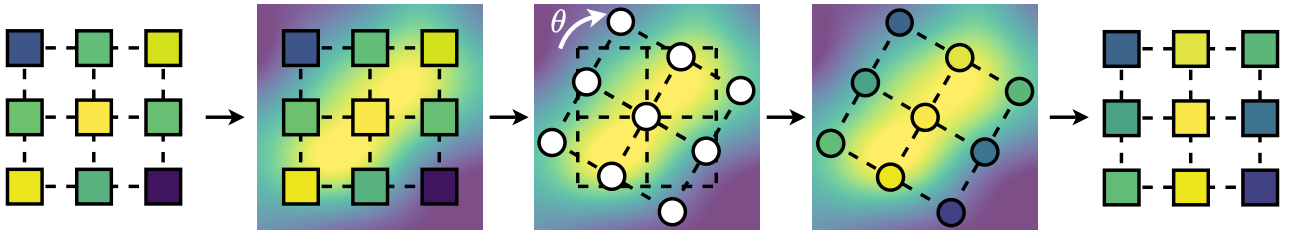
## 2. Related work

**Rotated object detection** attempts to extend generic horizontal detection to a finer-grained problem by introducing the oriented bounding boxes. Along this path, one line of work is devoted to establishing specialized rotated object detectors, including the feature refinement design in the detector neck [89, 86, 87], the oriented region proposal network [76, 7], the rotated region of interest (RoI) extraction mechanism [9, 76], detector head design [19, 33] and advanced label assignment strategy [52, 53]. Another line of work focuses on designing more flexible object representations. For example, Oriented RepPoints [44] represent objects as a set of sample points. Gliding Vertex [79] introduces a novel representation by adding four gliding offset variables to classical horizontal bounding box representation. CFA [17] models irregular object layout and shape as convex hulls. G-Rep [34] proposes a unified Gaussian representation to construct Gaussian distributions for oriented bounding box, quadrilateral bounding box, and point set. Meanwhile, a proper loss function for various oriented object representations has also been extensively studied. GWD [88] and KLD [90] convert the rotated bounding box into a 2-D Gaussian distribution, and then calculate the the Gaussian Wasserstein distance and Kullback-Leibler Divergence separately as losses. KFIoU [92] proposes an effective approximate SkeIoU loss on Gaussian modeling and Kalman filter. Furthermore, some studies approach the problem from unique perspectives. [85].

Albeit effective, the aforementioned works generally focus on the detector design, while the design of a proper backbone feature extractor is rarely explored. ReDet [20] incorporates rotation-equivariant operations [73] into the backbone to produce rotation-equivariant features. Although the orientation information is reserved by rotation-equivariant operations, the variation of oriented objects within an image and across the dataset is ignored. In contrast, the proposed adaptive rotated convolution method embraces this characteristic of rotated objects.

**Dynamic network** is an emerging research topic in the deep learning community [22]. In contrast to static network models, which share a fixed computation paradigm across different samples, dynamic networks can adapt their network structures or parameters to the input in the inference stage, and thus enjoy some favorable properties over static models such as efficiency, representation power, adaptiveness, compatibility, and interpretability. Dynamic networks can be divided into the following three categories: sample-wise, spatial-wise, and temporal-wise dynamic networks.

Sample-wise dynamic networks process different inputs in a data-dependent manner and are typically designed from two perspectives: dynamic architecture and dynamic parameter. The former one generally adjusts model architecture to allocate appropriate computation based on each sample, therefore reducing redundant computation for increased efficiency. Popular techniques include early-exiting[62, 5, 36, 71, 23, 21], layer skipping [69, 63], mixture of experts [58, 54, 12], and dynamic routing in supernets [61, 45]. In contrast, dynamic parameter approach adapts network parameters to every input with fixed com-




(a) A 3×3 conv kernel before rotation. (b) The original kernel spans a kernel space. (c) Rotate the axes to get the new sample coordinates. (d) Sample new parameters from the kernel space. (e) The conv kernel after rotation.

Figure 2. **The procedure of rotating a 3 × 3 convolution kernel.** (a) One channel of the original 3 × 3 convolution kernel. (b) By interpolation techniques, the 3×3 weight values can span into a 2D kernel space. (c) Rotate the original coordinates to get the sample coordinates for the new rotated convolution kernel. (d) Sample weight values in the new rotated coordinates from the kernel space. (e) The rotated convolution kernel weights are obtained by sampling from the original space.

putational graphs, with the goal of boosting the representation power with minimal increase of computational cost. The parameter adaptation can be achieved from four aspects: 1) adjusting the trained parameters based on the input [60, 81]; 2) directly generating the network parameters from the input [18, 39, 16, 51]; 3) adapting the kernel shape conditioned on the input [98, 8, 75]; and 4) rescaling the features with soft attention [66, 35, 26]. Spatial-wise dynamic networks perform spatially adaptive inference on the most informative regions, and reduce the unnecessary computation on less important areas. Existing works mainly include three levels of dynamic computation: resolution level [83, 97, 27, 28, 29], region level [72, 37] and pixel level [10, 64, 78, 24]. Temporal-wise dynamic networks extend the ideology of dynamic computation into the sequential data, typically on processing text data [25] and videos [70].

Our proposed method can be categorized into the parameter adjusting class. With adaptive rotated convolution kernel parameters, the proposed convolution module boosts the representation power of the backbone feature extractor, especially under the rotated object detection scenario.

## 3. Method

In this section, we first introduce the convolution kernel rotation mechanism given a rotation angle $\theta$ (Sec. 3.1). The network structure and the design methodology of the routing function are further presented (Sec. 3.2). Finally, the overall picture of the proposed adaptive convolution module is illustrated in Sec. 3.3. We also provide the implementation details of the ARC module in Sec. 3.4.

### 3.1. Rotate the convolution kernels

The standard convolution, adopted as the backbone of most oriented object detectors, employs consistent parameters to extract features from all image samples. In scenarios of rotated object detection, this implies that object instances, irrespective of their rotational angles, are processed using a static convolution kernel oriented in a fixed direction. To bridge the gap between arbitrarily-oriented object instances and these statically-oriented convolution kernels, we propose rotating the convolution kernels by sampling weights within the kernel space in a data-driven manner.

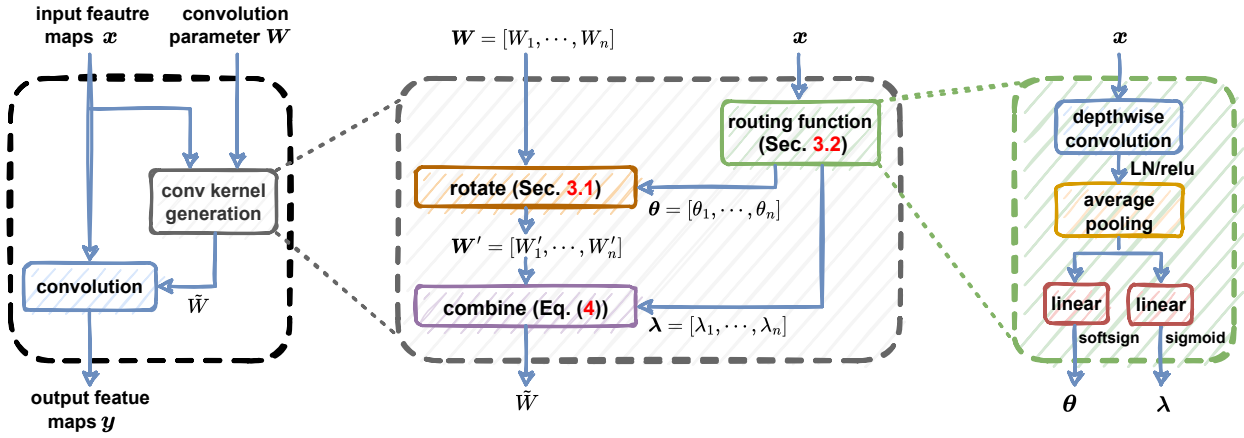
First, we illustrate how one channel of a convolution kernel rotates given a rotation angle $\theta$. We define the counter-clockwise direction as positive. Instead of treating the convolution weight as independent parameters (Fig. 2(a)), we treat them as sampled points from a kernel space. Therefore, the original convolution parameters can span a kernel space by interpolation (Fig. 2(b)). In practice, we use bilinear interpolation. The procedure of rotating a convolution kernel is the procedure of sampling new weight values in the rotated coordinates from the kernel space. The sample coordinates are obtained by revolving the original coordinates clockwise around the central point by $\theta$ degree (Fig. 2(c)). By sampling the values from the original kernel space in the rotated coordinates (Fig. 2(d)), the rotated convolution kernel is obtained (Fig. 2(e)). Note that to rotate the convolution kernel by $\theta$ degrees counter-clockwise, the coordinates in Fig. 2 need to take a $\theta$ degree clockwise rotation.

Now that the mechanism of rotating one channel of a convolution kernel (with a shape of $[k, k]$, $k$ is the kernel size) is given, the rotation procedure of the overall parameters of a convolution layer (with a shape of $[C_{out}, C_{in}, k, k]$, where $C_{in}$ and $C_{out}$ denote the number of input channels and the number of output channels, respectively) is easily extended. We simply apply the same procedure to all the $C_{in}$ channels and all the $C_{out}$ kernels to obtain the rotated weight parameter for a convolution layer.

### 3.2. Routing function

The routing function is one of the key components of the proposed adaptive rotated convolution module because it predicts the rotation angles and the combination weights in a data-dependent manner. The routing function takes the image feature $\boldsymbol{x}$ as input and predicts a set of rotated angles $[\theta_1, \cdots, \theta_n]$ for the set of kernels and the corresponding combination weights $[\lambda_1, \cdots, \lambda_n]$ for them.

The overall architecture of the routing function is shown




(a) Adaptive rotate convolution module (b) Kernel rotation and combination process (c) Routing function

Figure 3. **The illustration of the proposed adaptive rotated convolution module (ARC module).** (a) The macro view of the ARC module. The final convolution kernels $\tilde{W}$ are generated from the original convolution parameter $\boldsymbol{W}$, guided by the input feature map $\boldsymbol{x}$. (b) The process of convolution kernel generation. The kernels $\boldsymbol{W} = [W_1, \cdots, W_n]$ rotate $\boldsymbol{\theta} = [\theta_1, \cdots, \theta_n]$ degree and then combine together with the weight $\boldsymbol{\lambda} = [\lambda_1, \cdots, \lambda_n]$. The $\boldsymbol{\theta}$ and the $\boldsymbol{\lambda}$ are predicted by the routing function in a data-dependent manner. (c) The architecture of the routing function. The image feature $\boldsymbol{x}$ is encoded by a depthwise convolution with an average pooling layer followed behind. The $\boldsymbol{\theta}$ and the $\boldsymbol{\lambda}$ are predicted by two different branches with different activation functions, respectively.

in Fig. 3(c). The input image feature $\boldsymbol{x}$, with a size of $[C_{\text{in}}, H, W]$, is first fed into a lightweight depthwise convolution with a 3×3 kernel size, followed by a layer normalization [2] and a ReLU activation. Then the activated feature is averaged pooled into a feature vector with $C_{\text{in}}$ dimensions. The pooled feature vector is passed into two different branches. The first branch is the rotation angle prediction branch, composed of a linear layer and a softsign activation. We set the bias of this linear layer as false to avoid learning biased angles. The softsign activation is adopted to have a low saturation speed. In addition, the output of softsign layer is multiplied by a coefficient to enlarge the range of rotation. The second branch, named combination weights prediction branch, is responsible for predicting the combination weights $\boldsymbol{\lambda}$. It is constructed by a linear layer with bias and a sigmoid activation. The routing function is initialized from a zero-mean truncated normal distribution with 0.2 standard deviation to let the module produce small values at the start of the learning procedure.

### 3.3. Adaptive rotated convolution module

In a regular convolution layer, the same convolution kernel is used for all input images. In contrast, the convolution kernel is adaptively rotated according to different input feature maps in the proposed adaptive rotated convolution module. Considering that object instances in an image usually face multiple directions, we introduce a conditional computation mechanism to handle objects of multiple orientations in the ARC module. The ARC module has $n$ kernels $(W_1, \cdots, W_n)$, each with a shape of $[C_{out}, C_{\text{in}}, k, k]$. Given the input feature $\boldsymbol{x}$, the routing function $f$ predicts a set of rotation angles $\boldsymbol{\theta}$ and combination weights $\boldsymbol{\lambda}$:

$$\boldsymbol{\theta}, \boldsymbol{\lambda} = \boldsymbol{f}(\boldsymbol{x}). \tag{1}$$

The $n$ kernels first rotate individually according to the predicted rotation angle $\boldsymbol{\theta} = [\theta_1, \theta_2, \cdots, \theta_n]$,

$$W'_i = \text{Rotate}(W_i; \theta_i), i = 1, 2, \cdots, n, \tag{2}$$

where $\theta_i$ denotes the rotation angle for $W_i$, $W'_i$ is the rotated kernel, and Rotate$(\cdot)$ is the procedure described in Sec. 3.1. A naive usage of these rotated kernels is convolving them with the input feature maps separately and adding the output feature maps together in an element-wise manner

$$\boldsymbol{y} = \lambda_1(W'_1 * \boldsymbol{x}) + \lambda_2(W'_2 * \boldsymbol{x}) + \cdots + \lambda_n(W'_n * \boldsymbol{x}), \tag{3}$$

where $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \cdots, \lambda_n]$ is the combination weights predicted by the routing function, $*$ is the convolution operation, and $\boldsymbol{y}$ is the combined output feature maps. Inspired by the conditional parameterization technique [81], the above formula Eq. (3) can be written as

$$\boldsymbol{y} = (\lambda_1 W'_1 + \lambda_2 W'_2 + \cdots + \lambda_n W'_n) * \boldsymbol{x}, \tag{4}$$

which means convolving the input feature separately and adding the output of these features (Eq. (3)) is equivalent to performing one convolution operation with the combined convolution weight of these kernels (Eq. (4)). This strategy increases the representation ability of the network for capturing the features of multiple oriented objects yet remains highly efficient because the heavy convolution computation only occurs once in Eq. (4) compared to in Eq. (3).

| Method | Backbone | PL | BD | BR | GTF | SV | LV | SH | TC | BC | ST | SBF | RA | HA | SP | HC | mAP |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Single-stage methods* | | | | | | | | | | | | | | | | | |
| Rotated RetinaNet [47] | R50 | 89.29 | 78.54 | 41.13 | 66.29 | 76.92 | 61.68 | 77.40 | 90.89 | 81.37 | 82.89 | 59.05 | 63.79 | 55.05 | 61.95 | 40.14 | 68.42 |
| | **ARC-R50** | 89.59 | 82.45 | 41.66 | 71.30 | 77.71 | 63.15 | 78.04 | 90.90 | 84.95 | 83.55 | 57.74 | 69.06 | 54.72 | 72.34 | 54.58 | **71.45**$_{(\uparrow 3.03)}$ |
| R3Det [86] | R50 | 89.00 | 75.60 | 46.64 | 67.09 | 76.18 | 73.40 | 79.02 | 90.88 | 78.62 | 84.88 | 59.00 | 61.16 | 63.65 | 62.39 | 37.94 | 69.70 |
| | **ARC-R50** | 89.49 | 78.04 | 46.36 | 68.89 | 77.45 | 72.87 | 82.76 | 90.90 | 83.07 | 84.89 | 58.72 | 68.61 | 64.75 | 68.39 | 49.67 | **72.32**$_{(\uparrow 2.62)}$ |
| $S^2$ANet [19] | R50 | 89.30 | 80.11 | 50.97 | 73.91 | 78.59 | 77.34 | 86.38 | 90.91 | 85.14 | 84.84 | 60.45 | 66.94 | 66.78 | 68.55 | 51.65 | 74.13 |
| | **ARC-R50** | 89.28 | 78.77 | 53.00 | 72.44 | 79.81 | 77.84 | 86.81 | 90.88 | 84.27 | 86.20 | 60.74 | 68.97 | 66.35 | 71.25 | 65.77 | **75.49**$_{(\uparrow 1.36)}$ |
| *Two-stage methods* | | | | | | | | | | | | | | | | | |
| Rotated Faster R-CNN [57] | R50 | 89.40 | 81.81 | 47.28 | 67.44 | 73.96 | 73.12 | 85.03 | 90.90 | 85.15 | 84.90 | 56.60 | 64.77 | 64.70 | 70.28 | 62.22 | 73.17 |
| | **ARC-R50** | 89.49 | 82.11 | 51.02 | 70.38 | 79.07 | 75.06 | 86.18 | 90.91 | 84.23 | 86.41 | 56.10 | 69.42 | 65.87 | 71.90 | 63.47 | **74.77**$_{(\uparrow 1.60)}$ |
| CFA [17] | R50 | 88.85 | 75.31 | 50.68 | 68.27 | 79.83 | 74.61 | 86.43 | 90.85 | 80.67 | 85.11 | 50.29 | 61.05 | 64.99 | 67.00 | 16.65 | 69.37 |
| | **ARC-R50** | 88.96 | 79.77 | 52.08 | 75.32 | 79.46 | 74.79 | 86.98 | 90.87 | 80.90 | 86.07 | 58.16 | 66.85 | 66.15 | 69.90 | 46.74 | **73.53**$_{(\uparrow 4.16)}$ |
| Oriented R-CNN [76] | R50 | 89.48 | 82.59 | 54.42 | 72.58 | 79.01 | 82.43 | 88.26 | 90.90 | 86.90 | 84.34 | 60.79 | 67.08 | 74.28 | 69.77 | 54.27 | 75.81 |
| | **ARC-R50** | 89.40 | 82.48 | 55.33 | 73.88 | 79.37 | 84.05 | 88.06 | 90.90 | 86.44 | 84.83 | 63.63 | 70.32 | 74.29 | 71.91 | 65.43 | **77.35**$_{(\uparrow 1.54)}$ |

Table 1. **Experiment results on the DOTA dataset among various detectors**. In the backbone column, R50 stands for ResNet-50 [31], and ARC-R50 is the backbone network that replaces the $3 \times 3$ convolution in the last three stage of ResNet-50 with the proposed ARC module. We conduct experiments on a variety of popular rotated object detection networks, including both single-stage methods (Rotated RetinaNet [47], R3Det [86], $S^2$ANet [19] ) and two-stage approaches (Rotated FasterR-CNN [57], CFA [17], Oriented R-CNN [76]). Experimental results validate the effectiveness and compatibility of the proposed method on various oriented detectors.

| Method | Backbone | $AP_{50}$ | $AP_{75}$ | mAP |
|---|---|---|---|---|
| Rotated RetinaNet [47] | R50 | 84.20 | 58.50 | 52.70 |
| | **ARC-R50** | **85.10** | **60.20** | **53.97**$_{(\uparrow 1.27)}$ |
| $S^2$ANet [19] | R50 | 89.70 | 65.30 | 55.65 |
| | **ARC-R50** | **90.00** | **67.40** | **57.77**$_{(\uparrow 2.12)}$ |
| Oriented R-CNN [76] | R50 | 90.40 | 88.81 | 70.55 |
| | **ARC-R50** | **90.41** | **89.02** | **72.39**$_{(\uparrow 1.84)}$ |

Table 2. **Experiment results on the HRSC2016 dataset.** The proposed kernel rotation mechanism is also effective on the HRSC2016 dataset on some popular rotated object detectors.

### 3.4. Implementation details

Since the proposed ARC module can conveniently serve as a plug-and-play module for any backbone network with convolutional layer, we build the proposed backbone network ARC-ResNet based on the commonly used ResNet [31]. For the following experiments, we replace all the $3 \times 3$ convolutions in the last three stages and keep the $1 \times 1$ convolution the same as before, since $1 \times 1$ convolution is rotational invariant. The ablation studies on the performance of replacing different parts of the network are illustrated in Sec. 4.4. In addition, we scale down the learning rate of the proposed backbone network during training. This adjustment on the rotated object detection tasks helps avoid drastic changes in the predicted rotation angles.

## 4. Experiments

In this section, we empirically evaluate our proposed backbone network equipped with adaptive rotated convolution on various detection networks. We begin by introducing the detailed experiment setup, which includes datasets and training configurations, in Sec. 4.1. Then the main results of our method with various rotated object detectors on two commonly used datasets are presented in Sec. 4.2. The comparison results with competing approaches are demonstrated in Sec. 4.3. Finally, the ablation studies in Sec. 4.4 and the visualization results in Sec. 4.5 further validate the effectiveness of the proposed method.

### 4.1. Experiment settings

**Datasets.** We evaluate the proposed methods on two widely used oriented object detection benchmarks, *i.e.*, DOTA-v1.0 [74] and HRSC2016 [49].

DOTA [74] is a large-scale rotated object detection dataset containing 2806 photographs and 188282 instances with oriented bounding box annotations. The following fifteen object classes are covered in this dataset: Plane (PL), Baseball diamond (BD), Bridge (BR), Ground track field (GTF), Small vehicle (SV), Large vehicle (LV), Ship (SH), Tennis court (TC), Basketball court (BC), Storage tank (ST), Soccer-ball field (SBF), Roundabout (RA), Harbor (HA), Swimming pool (SP), and Helicopter (HC). The image size of the DOTA dataset is extensive: from $800 \times 800$ to $4000 \times 4000$ pixels. We crop the raw images into $1024 \times 1024$ patches with a stride of 824, which means the pixel overlap between two adjacent patches is 200. With regard to multi-scale training and testing, we first resize the raw pictures at three scales (0.5, 1.0, and 1.5) and crop them into $1024 \times 1024$ patches with the stride of 524. Following the common practice, we use both the training set and the validation set for training, and the testing set for testing.

| Method | Backbone | PL | BD | BR | GTF | SV | LV | SH | TC | BC | ST | SBF | RA | HA | SP | HC | mAP |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ***Single-stage methods*** | | | | | | | | | | | | | | | | | |
| DRN [55] | H104 | 88.91 | 80.22 | 43.52 | 63.35 | 73.48 | 70.69 | 84.94 | 90.14 | 83.85 | 84.11 | 50.12 | 58.41 | 67.62 | 68.60 | 52.50 | 70.70 |
| R3Det [86] | R101 | 88.76 | 83.09 | 50.91 | 67.27 | 76.23 | 80.39 | 86.72 | 90.78 | 84.68 | 83.24 | 61.98 | 61.35 | 66.91 | 70.63 | 53.94 | 73.79 |
| PIoU [6] | DLA34 | 80.90 | 69.70 | 24.10 | 60.20 | 38.30 | 64.40 | 64.80 | 90.90 | 77.20 | 70.40 | 46.50 | 37.10 | 57.10 | 61.90 | 64.00 | 60.50 |
| RSDet [56] | R101 | 89.80 | 82.90 | 48.60 | 65.20 | 69.50 | 70.10 | 70.20 | 90.50 | 85.60 | 83.40 | 62.50 | 63.90 | 65.60 | 67.20 | 68.00 | 72.20 |
| DAL [53] | R50 | 88.68 | 76.55 | 45.08 | 66.80 | 67.00 | 76.76 | 79.74 | 90.84 | 79.54 | 78.45 | 57.71 | 62.27 | 69.05 | 73.14 | 60.11 | 71.44 |
| $S^2$ANet [19] | R50 | 89.30 | 80.11 | 50.97 | 73.91 | 78.59 | 77.34 | 86.38 | 90.91 | 85.14 | 84.84 | 60.45 | 66.94 | 66.78 | 68.55 | 51.65 | 74.13 |
| G-Rep [34] | R101 | 88.89 | 74.62 | 43.92 | 70.24 | 67.26 | 67.26 | 79.80 | 90.87 | 84.46 | 78.47 | 54.59 | 62.60 | 66.67 | 67.98 | 52.16 | 70.59 |
| ***Two-stage methods*** | | | | | | | | | | | | | | | | | |
| ICN [1] | R101 | 81.36 | 74.30 | 47.70 | 70.32 | 64.89 | 67.82 | 69.98 | 90.76 | 79.06 | 78.20 | 53.64 | 62.90 | 67.02 | 64.17 | 50.23 | 68.16 |
| CAD-Net [94] | R101 | 87.80 | 82.40 | 49.40 | 73.50 | 71.10 | 63.50 | 76.60 | 90.90 | 79.20 | 73.30 | 48.40 | 60.90 | 62.00 | 67.00 | 62.20 | 69.90 |
| RoI Trans [9] | R101 | 88.64 | 78.52 | 43.44 | 75.92 | 68.81 | 73.68 | 83.59 | 90.74 | 77.27 | 81.46 | 58.39 | 53.54 | 62.83 | 58.93 | 47.67 | 69.56 |
| SCRDet [89] | R101 | 89.98 | 80.65 | 52.09 | 68.36 | 68.36 | 60.32 | 72.41 | 90.85 | 87.94 | 86.86 | 65.02 | 66.68 | 66.25 | 68.24 | 65.21 | 72.61 |
| G. Vertex [79] | R101 | 89.64 | 85.00 | 52.26 | 77.34 | 73.01 | 73.14 | 86.82 | 90.74 | 79.02 | 86.81 | 59.55 | 70.91 | 72.94 | 70.86 | 57.32 | 75.02 |
| FAOD [43] | R101 | 90.21 | 79.58 | 45.49 | 76.41 | 73.18 | 68.27 | 79.56 | 90.83 | 83.40 | 84.68 | 53.40 | 65.42 | 74.17 | 69.69 | 64.86 | 73.28 |
| CenterMap [68] | R50 | 88.88 | 81.24 | 53.15 | 60.65 | 78.62 | 66.55 | 78.10 | 88.83 | 77.80 | 83.61 | 49.36 | 66.19 | 72.10 | 72.36 | 58.70 | 71.74 |
| FR-Est [14] | R101 | 89.63 | 81.17 | 50.44 | 70.19 | 73.52 | 77.98 | 86.44 | 90.82 | 84.13 | 83.56 | 60.64 | 66.59 | 70.59 | 66.72 | 60.55 | 74.20 |
| Mask OBB [67] | R50 | 89.61 | 85.09 | 51.85 | 72.90 | 75.28 | 73.23 | 85.57 | 90.37 | 82.08 | 85.05 | 55.73 | 68.39 | 71.61 | 69.87 | 66.33 | 74.86 |
| ReDet [20] | ReR50 | 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 | 63.59 | 76.25 |
| AOPG [7] | R101 | 89.14 | 82.74 | 51.87 | 69.28 | 77.65 | 82.42 | 88.08 | 90.89 | 86.26 | 85.13 | 60.60 | 66.30 | 74.05 | 67.76 | 58.77 | 75.39 |
| SASM [33] | R50 | 86.42 | 78.97 | 52.47 | 69.84 | 77.30 | 75.99 | 86.72 | 90.89 | 82.63 | 85.66 | 60.13 | 68.25 | 73.98 | 72.22 | 62.37 | 74.92 |
| Oriented R-CNN [76] | R50 | 89.48 | 82.59 | 54.42 | 72.58 | 79.01 | 82.43 | 88.26 | 90.90 | 86.90 | 84.34 | 60.79 | 67.08 | 74.28 | 69.77 | 54.27 | 75.81 |
| | **ARC-R50** | 89.40 | 82.48 | 55.33 | 73.88 | 79.37 | 84.05 | 88.06 | 90.90 | 86.44 | 84.83 | 63.63 | 70.32 | 74.29 | 71.91 | 65.43 | **77.35**$_{(\uparrow 1.54)}$ |
| | R101 | 89.51 | 84.50 | 54.67 | 73.10 | 78.77 | 82.87 | 88.08 | 90.90 | 86.97 | 85.38 | 63.06 | 67.35 | 75.91 | 68.73 | 51.91 | 76.11 |
| | **ARC-R101** | 89.39 | 83.58 | 57.51 | 75.94 | 78.75 | 83.58 | 88.08 | 90.90 | 85.93 | 85.38 | 64.03 | 68.65 | 75.59 | 72.03 | 65.68 | **77.70**$_{(\uparrow 1.59)}$ |

Table 3. **Experimental results on the DOTA dataset compared with state-of-the-art methods.** In the backbone column, H104 denotes the 104-layer hourglass network [82], DLA34 refers to the 34-layer deep layer aggregation network [95], R50 and R101 stand for ResNet-50 and ResNet-101 [31], respectively, ReR50 is proposed in ReDet [20] with rotation-equivariant operations, ARC-R50 and ARC-R101 is the proposed backbone network which replaces the 3×3 convolutions in ResNets with the proposed adaptive rotated convolution.

| Method | Backbone | mAP |
|---|---|---|
| R3Det [86] | R152 | 76.47 |
| SASM [33] | RX101 | 79.17 |
| $S^2$ANet [19] | R50 | 79.42 |
| ReDet [20] | ReR50 | 80.10 |
| R3Det-GWD [88] | R152 | 80.19 |
| R3Det-KLD [90] | R152 | 80.63 |
| AOPG [7] | R50 | 80.66 |
| KFIoU [92] | Swin-T | 80.93 |
| RVSA [65] | ViTAE-B | 81.24 |
| Oriented R-CNN [76] | R50 | 80.62 |
| | **ARC-R50** | **81.77**$_{(\uparrow 1.15)}$ |

Table 4. **Experiment results on the DOTA dataset.** The results are obtained under ***multi-scale training and testing*** strategies.

The mean average precision and the average precision of each category are obtained by submitting the testing results to the official evaluation server of the DOTA dataset.

HRSC2016 [49] is another widely-used arbitrary-oriented object detection benchmark. It contains 1061 images with sizes ranging from 300×300 to 1500×900. Both the training set (436 images) and validation set (181 images) are used for training and the remaining for testing. For the evaluation metrics on the HRSC2016 [49], we report the COCO [48] style mean average precision (mAP) as well as the average precision under 0.5 and 0.75 threshold ($AP_{50}$ and $AP_{75}$). In the data pre-processing procedure, we do not change the aspect ratios of images.

**Implementation Details.** The results on DOTA and HRSC2016 are obtained using the MMRotate [96] toolbox, except that Oriented R-CNN [76] is implemented with OBBDetection codebase. On the DOTA dataset, we train all the models for 12 epochs. On the HRSC2016 dataset, the Rotated RetinaNet [47] is trained for 72 epochs, while $S^2$ANet [19] and Oriented R-CNN [76] are trained for 36 epochs. The detailed configuration of various detectors is provided in the supplementary material. Although the detailed training configuration varies among different detectors and different datasets, we keep it the same between the experiments using our proposed backbone networks and baseline backbone networks for fair comparisons.

## 4.2. Effectiveness on various architectures

We compare the backbone network equipped with our proposed ARC module with counterparts that use a canonical ResNet-50 [31]. The experiment results on DOTA [74] dataset and HRSC2016 [49] dataset are shown in Tab. 1 and Tab. 2, respectively. From the results, we find that our method significantly improves the generalization ability of various rotated object detection networks. On the most pop-

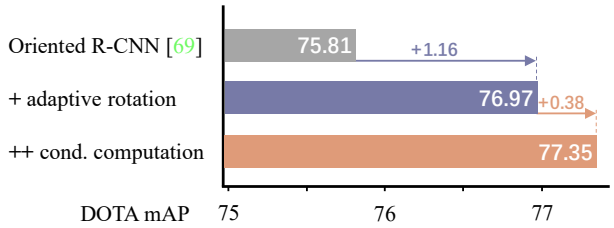



Figure 4. Ablation studies on the influence of adaptive kernel rotation and the effect of conditional computation. The experiments are conducted on Oriented R-CNN with DOTA dataset.

ular oriented object detection benchmark DOTA-v1.0, the proposed method achieves significant improvement on both single-stage and two-stage detectors.

For single-stage detectors, our method could improve 3.03% mAP for Rotated RetinaNet [47], 2.62% mAP for R3Det [86], and 1.36% mAP for $S^2$ANet [19]. For two-stage detectors, our method could also get more than 1.5% mAP improvement (+1.60% mAP for Rotated Faster R-CNN [57], +4.16% mAP for CFA [17] , and +1.54% mAP for Oriented R-CNN [76]). On the HRSC2016 [49] dataset, the proposed method could also achieve remarkable improvement (1.27% mAP improvement for Rotated RetinaNet [47], 2.12% for $S^2$ANet [19], and 1.84% for Oriented R-CNN [76]). These experimental results verify that the proposed backbone networks are compatible with various detection network architectures and can effectively improve their performance on the oriented object detection tasks.

### 4.3. Comparison with state-of-the-art methods.

We report full experimental results, including the average precision of each category and the mean average precision (mAP) on the DOTA dataset to make a fair comparison with the previous methods. We combine the proposed backbone network with one of the highly competitive method Oriented R-CNN [76]. The single-scale and the multi-scale training and testing results are shown in Tab. 3 and Tab. 4, respectively. When we use the ARC-ResNet-50 backbone, the adaptive rotated convolution can improve the mAP of Oriented R-CNN by 1.54% over the static convolution under the single scale training and testing strategy. As the depth of backbone network goes deeper to 101, the adaptive rotated convolution can still enhance the mAP by 1.59%. Under the multi-scale training and multi-scale testing strategy, our method reaches 81.77% mAP with ResNet-50 as the base model. This result is highly competitive and surpasses all other existing methods, even when compared with counterparts with vision transformer backbone [92, 65] or with advanced model pretraining mechanism [30, 65].

### 4.4. Ablation studies

We conduct ablation studies to analyze how different design choices affect the rotated object detection performance.

| Backbone | $n$ | Params(M) | FLOPs(G) | FPS (img/s) | mAP |
|---|---|---|---|---|---|
| R50 | 1 | 41.14 | 211.43 | 29.9 | 75.81 |
| R101 | 1 | 60.13 | 289.33 | 27.6 | 76.11 |
| ARC-R50 | 1 | 41.18 | 211.85 | 29.6 | 76.97 |
| ARC-R50 | 2 | 52.25 | 211.89 | 29.2 | 77.17 |
| ARC-R50 | 4 | 74.38 | 211.97 | 29.2 | 77.35 |
| ARC-R50 | 6 | 96.52 | 212.06 | 29.1 | 77.38 |

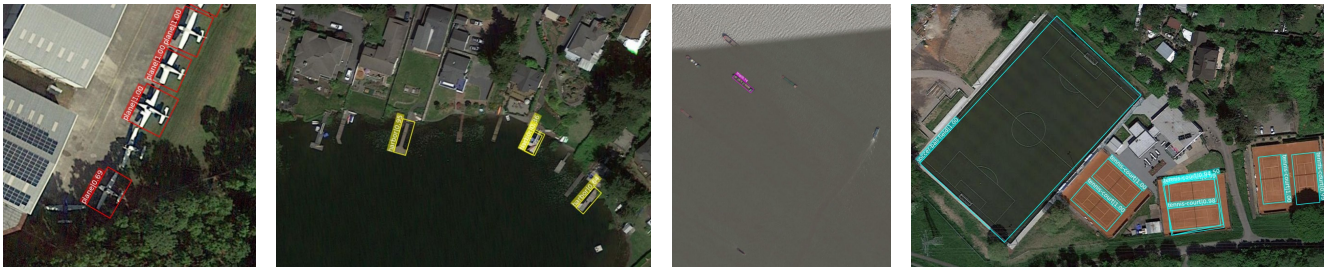
Table 5. Ablation studies on the kernel number $n$. The experiments are conducted on Oriented R-CNN with DOTA dataset.

We first demonstrate that the adaptive kernel rotation mechanism significantly outperforms the static convolution approach. We further present the effectiveness of the conditional computation mechanism and the ablation studies on kernel number $n$. We then examine the effect of replacing different stages of the backbone network. Finally, the architecture design of our routing function is studied.

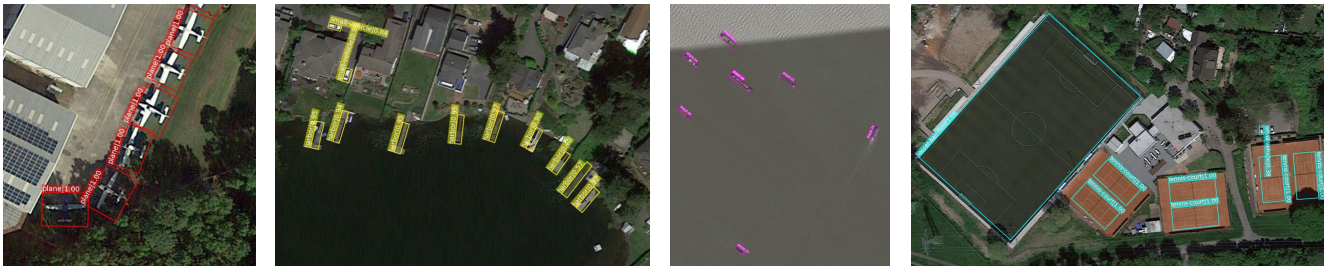
**Adaptive kernel rotation.** First, wecompare the performance between rotating the convolution kernels in a data-dependent manner and the static convolution on the DOTA dataset. The experiment results are shown in Fig. 4. From the comparison between the first two rows in Fig. 4, we find that by employing the adaptive kernel rotation, the performance of the object detector boosts (+1.16% mAP on Oriented R-CNN [76], which is highly competitive with SOTA methods). This experiment verifies the effectiveness of the proposed adaptive convolution kernel rotation approach for its adaptability on capturing oriented objects.

**Conditional computation.** The results in row 2 and row 3 of Fig. 4 show the effect of adopting more adaptive rotated kernels. By endowing the convolution kernel with more direction of rotation, the performance of the oriented object detector further increases. This is because the objects in an image usually have multiple orientations, and a single adaptive kernel is insufficient. Adopting more kernels with different orientation angles can endow the backbone feature extractor with more flexibility to produce high-quality features of these arbitrarily oriented objects.

**Ablation on kernel number.** Tab. 5 reports the information on the parameters, FLOPs, and inference speed (in FPS) and performance (in mAP) of different kernel numbers $n$. The FLOPs are calculated with $1024 \times 1024$ image resolution. The FPS is tested on an RTX 3090 with batch size 1, utilizing FP16 and torch.compile(). The results demonstrate that, as we progressively increase the number of kernels, the mAP exhibits a consistent upward trend. Meanwhile, FLOPs and FPS are essentially unchanged. This phenomenon demonstrate that the proposed method achieves remarkable improvement in mAP while maintaining a high level of efficiency, with only a marginal increase of 0.002% in FLOPs and less than 2.7% drop in FPS compared to the baseline model. Note that the number of parameters is no longer the bottleneck for evaluating model efficiency.



(a) Visualization results with baseline method



(b) Visualization results with our method

Figure 5. Visualization results of the oriented object detection on the test dataset of DOTA [74]. The upper row displays the bounding boxes predicted by Oriented R-CNN [76] with ResNet-50 backbone as baseline. Right column shows the predictions of the Oriented R-CNN equipped with our proposed ARC module. **Zoom in for best view.**

| Stage1 | Stage2 | Stage3 | Stage4 | mAP |
|---|---|---|---|---|
| - | - | - | - | 75.81 |
| - | - | - | ✓ | 77.17 (↑1.36) |
| - | - | ✓ | ✓ | 77.29 (↑1.48) |
| - | ✓ | ✓ | ✓ | **77.35** (↑1.54) |

Table 6. Ablation studies of the replacement strategy on the DOTA dataset. The experiments are conduct on Oriented R-CNN with ARC-ResNet-50 backbone network, which have 3, 4, 6, 3 blocks on each stage, respectively. The symbol '✓' means we replace all the convolution layers in this stage, while the symbol '-' means we do not replace any convolution blocks in this stage.

**Replacement strategy.** Since the feature pyramid network (FPN) [46] is attached to the last three stages of the backbone network, we do not replace the convolution layer in the first stage and ablate the replacement strategy in the last three stages. The ablation experiments are conducted on Oriented R-CNN [76] with a 50-depth backbone network on the DOTA dataset. Initially, we replace all the 3×3 convolution with the proposed ARC module in the last stage of the backbone network, and the mAP metric gets a 1.36% improvement over the baseline model. We further replace the convolution layer of more stages at the backbone network, and the performance on the oriented object detection benchmark improves steadily. As a result, we choose to replace all the last three stages in the backbone network.

**The structure of the routing function.** We ablate two designs in the routing function. The first design is spatial information encoding, which adds a depthwise convolution module before the average pooling layer (see Fig. 3(c)).

| adaptive combination | ✗ | ✗ | ✓ | ✓ |
|---|---|---|---|---|
| spatial info. encoding | ✗ | ✓ | ✗ | ✓ |
| mAP | 76.41 | 76.80 | 76.98 | **77.35** |

Table 7. Ablation studies on the structure of the routing function. *Adaptive combination* means using the combination weights prediction branch to adaptively combine the weight of each kernel, and *spatial encoding* refers to adding a DWConv-LN-ReLU module before the average pooling layer.

The second design uses the combination weights prediction branch to adaptively combine the weight of each kernel (the branch producing $\boldsymbol{\lambda}$ in Fig. 3(c)) rather than simply taking the average value of them. We conduct the experiments with Oriented R-CNN [76] on the DOTA dataset [74], and the corresponding experiment results are shown in Tab. 7. When we add the spatial encoding modules, the performance can increase from 76.41% to 76.80%. This is because the additional convolution layer helps the routing function to capture the spatial orientation information from the feature maps. Meanwhile, the introduction of the adaptive combination could also get a 0.47% reward in mAP, which shows the advantage of adopting the adaptiveness among different rotated kernels. When we use both of the designs, the oriented object detector achieves the highest performance. As a result, we choose to adopt both of the two designs into our proposed routing function.

### 4.5. Visualization

To give a deep understanding of our method, we visualize the predicted oriented bounding boxes and the corre-

sponding scores. The experiment is conducted with Oriented R-CNN [76] detector on the test set of DOTA. By comparing the results between the detector with our proposed backbone and that with the baseline backbone in Fig. 5, the proposed method shows its superiority. To be specific, thanks to its adaptiveness in handling arbitrarily oriented object instances, our method has a superior locating and identification ability on both small (*e.g.* ship in the third column), and median (*e.g.* plane and harbor in the first two columns), as well large (*e.g.* tennis court and soccer ball field in the last column) oriented object instances.

## 5. Conclusion

This paper proposed an adaptive rotated convolution module for rotated object detection. In the proposed approach, the convolution kernels rotate adaptively according to different object orientations in the images. An efficient conditional computation approach is further introduced to endow the network with more flexibility to capture the orientation information of multiple oriented objects within an image. The proposed module can be plugged into any backbone networks with convolution layer. Experiment results verify that, equipped with the proposed module in the backbone network, the performance of various oriented object detectors improves significantly on commonly used rotated object detection benchmarks while remaining efficient.

**Acknowledgement.** This work is supported in part by the National Key R&D Program of China under Grant 2021ZD0140407, the National Natural Science Foundation of China under Grants 62022048 and 62276150, THU-Bosch JCML and Guoqiang Institute of Tsinghua University. We also appreciate the generous donation of computing resources by High-Flyer AI.

## References

[1] Seyed Majid Azimi, Eleonora Vig, Reza Bahmanyar, Marco Körner, and Peter Reinartz. Towards multi-class object detection in unconstrained remote sensing imagery. In *ACCV*, 2018. 6

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv:1607.06450*, 2016. 4

[3] Ruzena Bajcsy, Yiannis Aloimonos, and John K Tsotsos. Revisiting active perception. *Autonomous Robots*, 2018. 1

[4] Jeannette Bohg, Karol Hausman, Bharath Sankaran, Oliver Brock, Danica Kragic, Stefan Schaal, and Gaurav S Sukhatme. Interactive perception: Leveraging action in perception and perception in action. *T-RO*, 2017. 1

[5] Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama. Adaptive neural networks for efficient inference. In *ICML*, 2017. 2

[6] Zhiming Chen, Kean Chen, Weiyao Lin, John See, Hui Yu, Yan Ke, and Cong Yang. Piou loss: Towards accurate oriented object detection in complex environments. In *ECCV*, 2020. 1, 6

[7] Gong Cheng, Jiabao Wang, Ke Li, Xingxing Xie, Chunbo Lang, Yanqing Yao, and Junwei Han. Anchor-free oriented proposal generator for object detection. *TGARS*, 2022. 1, 2, 6

[8] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *ICCV*, 2017. 3

[9] Jian Ding, Nan Xue, Yang Long, Gui-Song Xia, and Qikai Lu. Learning roi transformer for oriented object detection in aerial images. In *CVPR*, 2019. 1, 2, 6

[10] Xuanyi Dong, Junshi Huang, Yi Yang, and Shuicheng Yan. More is less: A more complicated network with less inference complexity. In *CVPR*, 2017. 3

[11] Jiafei Duan, Samson Yu, Hui Li Tan, Hongyuan Zhu, and Cheston Tan. A survey of embodied ai: From simulators to research tasks. *TETCI*, 2022. 1

[12] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *JMLR*, 2021. 2

[13] Stan Franklin. Autonomous agents as embodied ai. *Cybernetics & Systems*, 1997. 1

[14] Kun Fu, Zhonghan Chang, Yue Zhang, and Xian Sun. Point-based estimator for arbitrary-oriented object detection in aerial images. *TGARS*, 2020. 6

[15] Yasuhiro Fujita, Kota Uenishi, Avinash Ummadisingu, Prabhat Nagarajan, Shimpei Masuda, and Mario Ynocente Castro. Distributed reinforcement learning of targeted grasping with active vision for mobile manipulators. In *IROS*, 2020. 1

[16] Hang Gao, Xizhou Zhu, Steve Lin, and Jifeng Dai. Deformable kernels: Adapting effective receptive fields for object deformation. In *ICLR*, 2020. 3

[17] Zonghao Guo, Chang Liu, Xiaosong Zhang, Jianbin Jiao, Xiangyang Ji, and Qixiang Ye. Beyond bounding-box: Convex-hull feature adaptation for oriented and densely packed object detection. In *CVPR*, 2021. 1, 2, 5, 7

[18] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. In *ICLR*, 2017. 3

[19] Jiaming Han, Jian Ding, Jie Li, and Gui-Song Xia. Align deep features for oriented object detection. *TGARS*, 2021. 1, 2, 5, 6, 7

[20] Jiaming Han, Jian Ding, Nan Xue, and Gui-Song Xia. Redet: A rotation-equivariant detector for aerial object detection. In *CVPR*, 2021. 2, 6

[21] Yizeng Han, Dongchen Han, Zeyu Liu, Yulin Wang, Xuran Pan, Yifan Pu, Chao Deng, Junlan Feng, Shiji Song, and Gao Huang. Dynamic perceiver for efficient visual recognition. In *ICCV*, 2023. 2

[22] Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *TPAMI*, 2021. 2

[23] Yizeng Han, Yifan Pu, Zihang Lai, Chaofei Wang, Shiji Song, Junfen Cao, Wenhui Huang, Chao Deng, and Gao Huang. Learning to weight samples for dynamic early-exiting networks. In *ECCV*, 2022. 2

[24] Yizeng Han, Zhihang Yuan, Yifan Pu, Chenhao Xue, Shiji Song, Guangyu Sun, and Gao Huang. Latency-aware spatial-wise dynamic networks. In *NeurIPS*, 2022. 3

[25] Christian Hansen, Casper Hansen, Stephen Alstrup, Jakob Grue Simonsen, and Christina Lioma. Neural speed reading with structural-jump-lstm. In *ICLR*, 2019. 3

[26] Chunming He, Kai Li, Guoxia Xu, Jiangpeng Yan, Longxiang Tang, Yulun Zhang, Xiu Li, and Yaowei Wang. Hqg-net: Unpaired medical image enhancement with high-quality guidance. *arXiv preprint arXiv:2307.07829*, 2023. 3

[27] Chunming He, Kai Li, Yachao Zhang, Longxiang Tang, Yulun Zhang, Zhenhua Guo, and Xiu Li. Camouflaged object detection with feature decomposition and edge reconstruction. In *CVPR*, 2023. 3

[28] Chunming He, Kai Li, Yachao Zhang, Guoxia Xu, Longxiang Tang, Yulun Zhang, Zhenhua Guo, and Xiu Li. Weakly-supervised concealed object segmentation with sam-based pseudo labeling and multi-scale feature grouping. *arXiv preprint arXiv:2305.11003*, 2023. 3

[29] Chunming He, Kai Li, Yachao Zhang, Yulun Zhang, Zhenhua Guo, Xiu Li, Martin Danelljan, and Fisher Yu. Strategic preys make acute predators: Enhancing camouflaged object detectors by generating camouflaged objects. *arXiv preprint arXiv:2308.03166*, 2023. 3

[30] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*. 7

[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 5, 6

[32] Geremy Heitz and Daphne Koller. Learning spatial context: Using stuff to find things. In *ECCV*, 2008. 1

[33] Liping Hou, Ke Lu, Jian Xue, and Yuqiu Li. Shape-adaptive selection and measurement for oriented object detection. In *AAAI*, 2022. 1, 2, 6

[34] Liping Hou, Ke Lu, Xue Yang, Yuqiu Li, and Jian Xue. G-rep: Gaussian representation for arbitrary-oriented object detection. *arXiv:2205.11796*, 2022. 1, 2, 6

[35] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018. 3

[36] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens Van Der Maaten, and Kilian Q Weinberger. Multi-scale dense networks for resource efficient image classification. In *ICLR*, 2018. 2

[37] Gao Huang, Yulin Wang, Kangchen Lv, Haojun Jiang, Wenhui Huang, Pengfei Qi, and Shiji Song. Glance and focus networks for dynamic visual recognition. *IEEE TPAMI*, 2022. 3

[38] Vidit Jain and Erik Learned-Miller. Fddb: A benchmark for face detection in unconstrained settings. Technical report, University of Massachusetts, Amherst, 2010. 1

[39] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. In *NeurIPS*, 2016. 3

[40] Dimosthenis Karatzas, Lluis Gomez-Bigorda, Anguelos Nicolaou, Suman Ghosh, Andrew Bagdanov, Masakazu Iwamura, Jiri Matas, Lukas Neumann, Vijay Ramaseshan Chandrasekhar, Shijian Lu, et al. Icdar 2015 competition on robust reading. In *ICDAR*, 2015. 1

[41] Dimosthenis Karatzas, Faisal Shafait, Seiichi Uchida, Masakazu Iwamura, Lluis Gomez i Bigorda, Sergi Robles Mestre, Joan Mas, David Fernandez Mota, Jon Almazan Almazan, and Lluis Pere De Las Heras. Icdar 2013 robust reading competition. In *ICDAR*, 2013. 1

[42] Soomin Lee, Le Chen, Jiahao Wang, Alexander Liniger, Suryansh Kumar, and Fisher Yu. Uncertainty guided policy for active robotic 3d reconstruction using neural radiance fields. *RA-L*, 2022. 1

[43] Chengzheng Li, Chunyan Xu, Zhen Cui, Dan Wang, Tong Zhang, and Jian Yang. Feature-attentioned object detection in remote sensing imagery. In *ICIP*, 2019. 6

[44] Wentong Li, Yijie Chen, Kaixuan Hu, and Jianke Zhu. Oriented reppoints for aerial object detection. In *CVPR*, 2022. 1, 2

[45] Yanwei Li, Lin Song, Yukang Chen, Zeming Li, Xiangyu Zhang, Xingang Wang, and Jian Sun. Learning dynamic routing for semantic segmentation. In *CVPR*, 2020. 2

[46] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 8

[47] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *CVPR*, 2017. 5, 6, 7

[48] 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*, 2014. 6

[49] Zikun Liu, Hongzhen Wang, Lubin Weng, and Yiping Yang. Ship rotated bounding box space for ship extraction from high-resolution optical satellite images with complex backgrounds. *GRSL*, 2016. 1, 2, 5, 6, 7

[50] Kangchen Lv, Mingrui Yu, Yifan Pu, Xin Jiang, Gao Huang, and Xiang Li. Learning to estimate 3-d states of deformable linear objects from single-frame occluded point clouds. In *ICRA*, 2023. 1

[51] Ningning Ma, Xiangyu Zhang, Jiawei Huang, and Jian Sun. Weightnet: Revisiting the design space of weight networks. In *ECCV*, 2020. 3

[52] Qi Ming, Lingjuan Miao, Zhiqiang Zhou, Junjie Song, and Xue Yang. Sparse label assignment for oriented object detection in aerial images. *Remote Sensing*, 2021. 1, 2

[53] Qi Ming, Zhiqiang Zhou, Lingjuan Miao, Hongwei Zhang, and Linhao Li. Dynamic anchor learning for arbitrary-oriented object detection. In *AAAI*, 2021. 1, 2, 6

[54] Ravi Teja Mullapudi, William R Mark, Noam Shazeer, and Kayvon Fatahalian. Hydranets: Specialized dynamic architectures for efficient inference. In *CVPR*, 2018. 2

[55] Xingjia Pan, Yuqiang Ren, Kekai Sheng, Weiming Dong, Haolei Yuan, Xiaowei Guo, Chongyang Ma, and Changsheng Xu. Dynamic refinement network for oriented and densely packed object detection. In *CVPR*, 2020. 6

[56] Wen Qian, Xue Yang, Silong Peng, Junchi Yan, and Yue Guo. Learning modulated loss for rotated object detection. In *AAAI*, 2021. 1, 6

[57] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015. 5, 7

[58] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *ICLR*, 2017. 2

[59] Juil Sock, Guillermo Garcia-Hernando, and Tae-Kyun Kim. Active 6d multi-object pose estimation in cluttered scenarios with deep reinforcement learning. In *IROS*, 2020. 1

[60] Hang Su, Varun Jampani, Deqing Sun, Orazio Gallo, Erik Learned-Miller, and Jan Kautz. Pixel-adaptive convolutional neural networks. In *CVPR*, 2019. 3

[61] Ryutaro Tanno, Kai Arulkumaran, Daniel Alexander, Antonio Criminisi, and Aditya Nori. Adaptive neural trees. In *ICML*, 2019. 2

[62] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *ICPR*, 2016. 2

[63] Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. In *ECCV*, 2018. 2

[64] Thomas Verelst and Tinne Tuytelaars. Dynamic convolutions: Exploiting spatial sparsity for faster inference. In *CVPR*, 2020. 3

[65] Di Wang, Qiming Zhang, Yufei Xu, Jing Zhang, Bo Du, Dacheng Tao, and Liangpei Zhang. Advancing plain vision transformer towards remote sensing foundation model. *TGARS*, 2022. 6, 7

[66] Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. Residual attention network for image classification. In *CVPR*, 2017. 3

[67] Jinwang Wang, Jian Ding, Haowen Guo, Wensheng Cheng, Ting Pan, and Wen Yang. Mask obb: A semantic attention-based mask oriented bounding box representation for multi-category object detection in aerial images. *Remote Sensing*, 2019. 6

[68] Jinwang Wang, Wen Yang, Heng-Chao Li, Haijian Zhang, and Gui-Song Xia. Learning center probability map for detecting objects in aerial images. *TGARS*, 2020. 6

[69] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *ECCV*, 2018. 2

[70] Yulin Wang, Zhaoxi Chen, Haojun Jiang, Shiji Song, Yizeng Han, and Gao Huang. Adaptive focus for efficient video recognition. In *ICCV*, 2021. 3

[71] Yulin Wang, Rui Huang, Shiji Song, Zeyi Huang, and Gao Huang. Not all images are worth 16x16 words: Dynamic transformers for efficient image recognition. In *NeurIPS*, 2021. 2

[72] Yulin Wang, Kangchen Lv, Rui Huang, Shiji Song, Le Yang, and Gao Huang. Glance and focus: a dynamic approach to reducing spatial redundancy in image classification. In *NeurIPS*, 2020. 3

[73] Maurice Weiler and Gabriele Cesa. General e (2)-equivariant steerable cnns. In *NeurIPS*, 2019. 2

[74] Gui-Song Xia, Xiang Bai, Jian Ding, Zhen Zhu, Serge Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, and Liangpei Zhang. Dota: A large-scale dataset for object detection in aerial images. In *CVPR*, 2018. 1, 2, 5, 6, 8

[75] Zhuofan Xia, Xuran Pan, Shiji Song, Li Erran Li, and Gao Huang. Vision transformer with deformable attention. In *CVPR*, 2022. 3

[76] Xingxing Xie, Gong Cheng, Jiabao Wang, Xiwen Yao, and Junwei Han. Oriented r-cnn for object detection. In *ICCV*, 2021. 1, 2, 5, 6, 7, 8, 9

[77] Yuhan Xie, Minghao Lu, Rui Peng, and Peng Lu. Learning agile flights through narrow gaps with varying angles using onboard sensing. *RAL*, 2023. 1

[78] Zhenda Xie, Zheng Zhang, Xizhou Zhu, Gao Huang, and Stephen Lin. Spatially adaptive inference with stochastic feature sampling and interpolation. In *ECCV*, 2020. 3

[79] Yongchao Xu, Mingtao Fu, Qimeng Wang, Yukang Wang, Kai Chen, Gui-Song Xia, and Xiang Bai. Gliding vertex on the horizontal bounding box for multi-oriented object detection. *TPAMI*, 2020. 1, 2, 6

[80] Xiangjie Yan, Chen Chen, and Xiang Li. Adaptive vision-based control of redundant robots with null-space interaction for human-robot collaboration. In *ICRA*, 2022. 1

[81] Brandon Yang, Gabriel Bender, Quoc V Le, and Jiquan Ngiam. Condconv: Conditionally parameterized convolutions for efficient inference. In *NeurIPS*, 2019. 3, 4

[82] Jing Yang, Qingshan Liu, and Kaihua Zhang. Stacked hourglass network for robust facial landmark localisation. In *CVPRW*, 2017. 6

[83] Le Yang, Yizeng Han, Xi Chen, Shiji Song, Jifeng Dai, and Gao Huang. Resolution adaptive networks for efficient inference. In *CVPR*, 2020. 3

[84] Shuo Yang, Ping Luo, Chen-Change Loy, and Xiaoou Tang. Wider face: A face detection benchmark. In *CVPR*, 2016. 1

[85] Xue Yang and Junchi Yan. Arbitrary-oriented object detection with circular smooth label. In *ECCV*, 2020. 2

[86] Xue Yang, Junchi Yan, Ziming Feng, and Tao He. R3det: Refined single-stage detector with feature refinement for rotating object. In *AAAI*, 2021. 1, 2, 5, 6, 7

[87] Xue Yang, Junchi Yan, Wenlong Liao, Xiaokang Yang, Jin Tang, and Tao He. Scrdet++: Detecting small, cluttered and rotated objects via instance-level feature denoising and rotation loss smoothing. *TPAMI*, 2022. 1, 2

[88] Xue Yang, Junchi Yan, Qi Ming, Wentao Wang, Xiaopeng Zhang, and Qi Tian. Rethinking rotated object detection with gaussian wasserstein distance loss. In *ICML*, 2021. 1, 2, 6

[89] Xue Yang, Jirui Yang, Junchi Yan, Yue Zhang, Tengfei Zhang, Zhi Guo, Xian Sun, and Kun Fu. Scrdet: Towards more robust detection for small, cluttered and rotated objects. In *ICCV*, 2019. 1, 2, 6

[90] Xue Yang, Xiaojiang Yang, Jirui Yang, Qi Ming, Wentao Wang, Qi Tian, and Junchi Yan. Learning high-precision bounding box for rotated object detection via kullback-leibler divergence. In *NeurIPS*, 2021. 1, 2, 6

[91] Xue Yang, Gefan Zhang, Xiaojiang Yang, Yue Zhou, Wentao Wang, Jin Tang, Tao He, and Junchi Yan. Detecting rotated objects as gaussian distributions and its 3-d generalization. *TPAMI*, 2022. 1

[92] Xue Yang, Yue Zhou, Gefan Zhang, Jitui Yang, Wentao Wang, Junchi Yan, Xiaopeng Zhang, and Qi Tian. The kfiou loss for rotated object detection. In *ICLR*, 2023. 1, 2, 6, 7

[93] Cong Yao, Xiang Bai, Wenyu Liu, Yi Ma, and Zhuowen Tu. Detecting texts of arbitrary orientations in natural images. In *CVPR*, 2012. 1

[94] Gongjie Zhang, Shijian Lu, and Wei Zhang. Cad-net: A context-aware detection network for objects in remote sensing imagery. *TGARS*, 2019. 6

[95] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv:1904.07850*, 2019. 6

[96] Yue Zhou, Xue Yang, Gefan Zhang, Jiabao Wang, Yanyi Liu, Liping Hou, Xue Jiang, Xingzhao Liu, Junchi Yan, Chengqi Lyu, Wenwei Zhang, and Kai Chen. Mmrotate: A rotated object detection benchmark using pytorch. In *ACM MM*, 2022. 6

[97] Mingjian Zhu, Kai Han, Enhua Wu, Qiulin Zhang, Ying Nie, Zhenzhong Lan, and Yunhe Wang. Dynamic resolution network. In *NeurIPS*, 2021. 3

[98] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *CVPR*, 2019. 3