

AKConv: Convolutional Kernel with Arbitrary Sampled Shapes and Arbitrary Number of Parameters

Xin Zhang¹, Yingze Song¹, Tingting Song^{1,*}, Degang Yang^{1,2,*}, Yichen Ye³, Jie Zhou¹, and Liming Zhang¹

¹ College of Computer and Information Science, Chongqing Normal University

² Chongqing Engineering Research Center of Educational Big Data Intelligent Perception and Application

³ College of Electronic and Information Engineering, Southwest University

*Corresponding author: E-mail:address:{ttsong,yangdg}@cqnu.edu.cn

Abstract. Neural networks based on convolutional operations have achieved remarkable results in the field of deep learning, but there are two inherent flaws in standard convolutional operations. On the one hand, the convolution operation be confined to a local window and cannot capture information from other locations, and its sampled shapes is fixed. On the other hand, the size of the convolutional kernel is fixed to $k \times k$, which is a fixed square shape, and the number of parameters tends to grow squarely with size. It is obvious that the shape and size of targets are various in different datasets and at different locations. Convolutional kernels with fixed sample shapes and squares do not adapt well to changing targets. In response to the above questions, the Alterable Kernel Convolution (AKConv) is explored in this work, which gives the convolution kernel an arbitrary number of parameters and arbitrary sampled shapes to provide richer options for the trade-off between network overhead and performance. In AKConv, we define initial positions for convolutional kernels of arbitrary size by means of a new coordinate generation algorithm. To adapt to changes for targets, we introduce offsets to adjust the shape of the samples at each position. Moreover, we explore the effect of the neural network by using the AKConv with the same size and different initial sampled shapes. AKConv completes the process of efficient feature extraction by irregular convolutional operations and brings more exploration options for convolutional sampling shapes. Object detection experiments on representative datasets COCO2017, VOC 7+12 and VisDrone-DET2021 fully demonstrate the advantages of AKConv. AKConv can be used as a plug-and-play convolutional operation to replace convolutional operations to improve network performance. The code for the relevant tasks can be found at <https://github.com/CV-ZhangXin/AKConv>.

Introduction

Convolutional Neural Networks (CNNs), such as ResNet [1], DenseNet [2], and YOLO [3], have demonstrated excellent performance in various applications and have led the technological progress in many aspects of modern society. It has become indispensable from image recognition in self-driving cars [4] and medical image analysis [5] to intelligent surveillance [6] and personalized recommendation systems [7]. These successful network models rely heavily on convolutional operations, which efficiently extract local features in images and ensure model complexity.

Despite the fact that CNNs have achieved many successes in classification [8], object detection [9], semantic segmentation [10], etc., they still have some limitations. One of the most notable limitations concerns the choice of convolutional sample shape and size. Standard convolution operations tend to rely on square kernels with fixed sampling locations, such as 1×1 , 3×3 , 5×5 and 7×7 , etc. The sampling position of the regular kernel is not deformable and cannot be dynamically changed in response to changes in the shape of the object. Deformable Conv [11, 12] enhances network performance with offset to flexibly adjust the sampling shape of

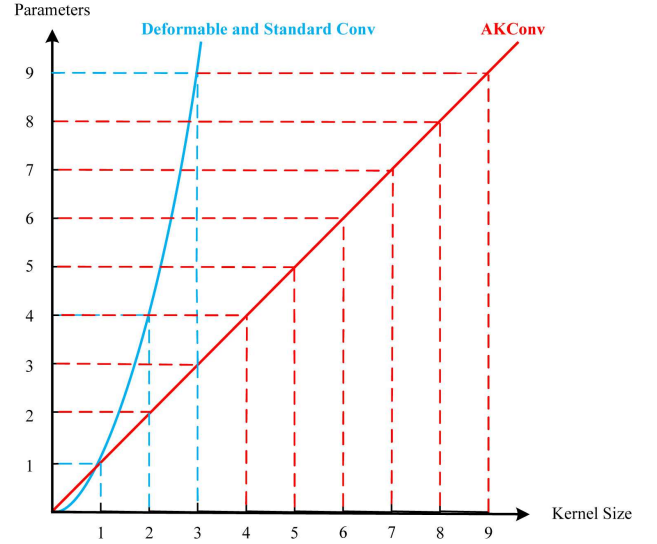


Fig. 1. Trend of the number of convolution parameters with increasing convolution size. It is evident that AKConv has more options compared to Deformable and standard Conv and the number of convolutional parameters shows a linear increase with the convolutional kernel size. To facilitate the description, we ignore the number of parameters for Deformable Conv and AKConv to learn offsets, as it is much smaller than the number of convolutional parameters involved in feature extraction.

the convolution kernel, which adapts to the change of the target. For instance, in [13, 14, 15], they utilized it to align features. Zhao et al. [16] improved the effectively of detection the dead fish by adding it in YOLOv4 [17]. Yang et al. [18] improved the YOLOv8 [19] for detecting the cattle by adding it in backbone. Li et al. [20] introduced Deformable Conv into deep image compression tasks [21, 22] to obtain content-adaptive receptive-fields.

Although the studies mentioned above have demonstrated the superior benefits of Deformable Conv. It is still not flexible enough. Because the convolution kernel is still limited to select kernel-size, and the number of convolution kernel parameters in standard convolutional operations and Deformable Conv shows a squared growth trend with the increase of the convolution kernel size, which is not a friendly way of growth to the hardware environment. Therefore, after careful analysis of standard convolution operations and Deformable Conv, we propose Alterable Kernel Convolution (AKConv). Unlike standard regular convolution, AKConv is a novel convolutional operations, which can extract features using efficient convolution kernels with any number of parameters such as (1, 2, 3, 4, 5, 6, 7...), which is not implemented by standard convolution and Deformable Convolution. AKConv can easily be used to replace the standard convolutional operations in a

network to improve network performance. Importantly, AKConv allows the number of convolutional parameters to trend linearly up or down, which is beneficial to hardware environments, and it can be used as an alternative to lightweight models to reduce the number of model parameters and computational overhead. Secondly, it has more options to improve the network performance in large kernels with sufficient resources. Fig. 1 shows that the regular convolutional kernel makes the number of parameters to show a square increasing trend, while AKConv only shows a linear increasing trend. Compared to the square growth trend, AKConv grows gently and provides more options for the choice of convolution kernel. Furthermore, its ideas can be extended to specific areas. Because, the special sampled shapes can be created for convolution operations according to the prior knowledge, and then dynamically and automatically adapt to changes in the target shape via offset. Object detection experiments on representative datasets VOC [23], COCO2017 [24], VisDrone-DET2021 [25] fully demonstrate the advantages of AKConv. In summary, our contributions are as follows:

1. For different sizes of convolutional kernels, we propose an algorithm to generate initial sampled coordinate for convolutional kernels of arbitrary sizes.
2. To adapt to the different variations of the target, we adjust the sampling position of the irregular convolutional kernel by the obtained offsets.
3. Compared to regular convolution kernels, the proposed AKConv realizes the function of irregular convolution kernels to extract features, providing convolution kernels with arbitrary sampling shapes and sizes for a variety of varying targets, which makes up for the shortcomings of regular convolutions.

2 Related works

In recent years, many works have considered and analyzed standard convolutional operations from different perspectives, and then designed novel convolutional operations to improve network performance.

Li et al. [26] argued that convolutional kernels sharing parameters across all spatial locations, which leads to limited modeling capabilities across different spatial locations, and do not effectively capture spatially long-range relationships. Secondly, the approach of using a different convolution kernel for each output channel is actually not efficient. Therefore, to address these shortcomings, they proposed the Involution operator, which inverts the features of the convolutional operation to improve network performance. Qi et al. [27] proposed the DSConv based on Deformable Conv. The offset obtained from learning in Deformable Conv is freedom, leading to the model losing a small percentage of fine structure features, which poses a great challenge for the task of segmenting elongated tubular structures, therefore, they proposed the DSConv. Zhang et al. [28] understood the spatial attention mechanism from a new perspective, they asserted that the spatial attention mechanism essentially solves the problem of parameter sharing of convolutional operations. However, some spatial attention mechanisms, such as CBAM [29] and CA [30], not completely solve the problem of large-size convolutional parameter sharing. Therefore, they proposed RFA-Conv. Chen et al. [31] proposed the Dynamic Conv. Unlike using a convolutional kernel for every layers, the Dynamic

Conv dynamically aggregated multiple parallel convolutional kernels based on their attention. The Dynamic Conv provided greater representation of features. Tan et al. [32] argued that kernel size is often neglected in CNNs, which may affect the accuracy and efficiency of the network. Second, using only layer-by-layer convolution does not utilize the full potential of convolutional networks. Therefore, they proposed MixConv, which naturally mixes multiple kernel sizes in a single convolution to improve performance of networks.

Although these methods improve the performance of convolutional operations, they are still limited to regular convolutional operations and do not allow multiple variations of convolutional sample shapes. In contrast, our proposed AKConv can efficiently extract features using a convolutional kernel with arbitrary number of parameters and sample shapes.

3 Methods

3.1 Define the initial sampling position

Convolutional neural networks are based on the convolution operation, which localizes the features at the corresponding locations by means of a regular sampling grid. In [11, 33, 34], the regular sampling grid for the 3×3 convolution operation is given. Let R denote the sampling grid, then R is denoted as follows:

$$R = \{(-1, -1), (-1, 0), \dots, (0, 1), (1, 1)\} \quad (1)$$

However, the sampling grid is regular, while AKConv targets irregularly shaped convolutional kernels. Therefore, to allow irregular convolutional kernels to have a sampling grid, we create an algorithm for arbitrary size convolution, which generates the initial sampling coordinates of the convolutional kernel P_n . First, we generate the sampling grid as a regular sampling grid, then the irregular grids is created for the remaining sampling points, and finally, we stitch them to generate the overall sampling grid. The pseudo code is as in Algorithm 1.

As shown in Fig. 2, it shown that the initial sampled coordinates is generated for arbitrary size convolution. The sampling grid of the regular convolution is centered at the $(0, 0)$ point. While the irregular convolution has no center at many sizes, to adapt to the size of the convolution used, we set the upper left corner $(0, 0)$ point as the sampling origin in the algorithm.

After defining the initial coordinates P_n for the irregular convolution, the corresponding convolution operation at position P_0 can be defined as follows:

$$Conv(P_0) = \sum w \times (P_0 + P_n) \quad (2)$$

Here, w denotes the convolutional parameter. However, the irregular convolution operations are impossible to realize, because irregular sampling coordinates cannot be matched to the corresponding size convolution operations, e.g., convolution of sizes 5, 7, and 13. Cleverly, our proposed AKConv realizes it.

3.2 Alterable convolutional operation

It is obvious that the standard convolutional sampling position is fixed, which leads to the convolution can only extract the local information of the current window, and can

Algorithm 1 Pseudo-code for initial coordinate generation for convolution kernel in a PyTorch-like.

```

# func get_p_n(num_param, dtype)
# num_param: the kernel size of AKConv
# dtype: the type of data

##### function body #####
# get a base integer to define coordinate
base_int = round(math.sqrt(num_param))
row_number = num_param // base_int
mod_number = num_param % base_int

# get the sampled coordinate of regular kernels

p_n_x, p_n_y = torch.meshgrid(
    torch.meshgrid(0, row_number),
    torch.meshgrid(0, base_int))

# flatten the sampled coordinate of regular kernels
p_n_x = torch.flatten(p_n_x)
p_n_y = torch.flatten(p_n_y)

# get the sampled coordinate of irregular kernels
If mod_number > 0:
    mod_p_n_x, mod_p_n_y = torch.meshgrid(
        torch.arange(row_number, row_number + 1),
        torch.arange(0, mod_number))

    mod_p_n_x = torch.flatten(mod_p_n_x)
    mod_p_n_y = torch.flatten(mod_p_n_y)
    P_n_x, p_n_y = torch.cat((p_n_x, mod_p_n_x)), torch.cat((
        p_n_y, mod_p_n_y))

# get the completed sampled coordinate
p_n = torch.cat([p_n_x, p_n_y], 0)
p_n = p_n.view(1, 2 * num_param, 1, 1).type(dtype)
return p_n

```

not capture the information of other positions. Deformable Conv learns the offsets through convolutional operations to adjust the sampling grid of the initial regular pattern. The approach compensates for the shortcomings of the convolution operation to a certain extent. However, the standard convolution and Deformable Conv are regular sampling grids that not allow convolution kernels with arbitrary number of parameters. Moreover, as the size of the convolution kernel increases their number of convolution parameters tends to increase by a square, which is not friendly for the hardware environment. Therefore, we propose a novel Alterable convolutional operation (AKConv). As shown in Fig. 3, it illustrates the overall structure of an AKConv of size 5.

Similar to Deformable Conv, in AKConv, the offset of the corresponding kernel are first obtained by convolution operations, which has the dimensions $(B, 2N, H, W)$, where N is the convolution kernel size. Take Fig. 3 as an example, $N = 5$. Then the modified coordinates are obtained by summing offset and original coordinates $(P_0 + P_n)$. Finally the features at the corresponding positions are obtained by interpolating and resampling. It is difficult to extract the features corresponding to the sampled positions of the irregular convolution kernel. To solve this problem, we found that there are many ways to solve it after deep thinking. In Deformable Conv [11] and RFACnv [28], they stack the 3×3 convolutional features in spatial dimensions. Then, a convolution operation with a step size of 3 is used to extract the features. However, this method targets square sampling shapes. Therefore, the features can be stacked on rows or columns to use the column convolution or row convolution to extract features corresponding to irregular sampling shapes. The features are extracted to use a convolutional kernel of the appropriate size and step size. Moreover, we can transform the features into four dimensions (C, N, H, W) , and then use Conv3d with step size and convolution size $(N, 1, 1)$ to extract the features. Of course, we can also stack the features on the channel dimension to (CN, H, W) , and then use 1×1 convolution to reduce the dimension to (C, H, W) . So all these methods mentioned above can ex-

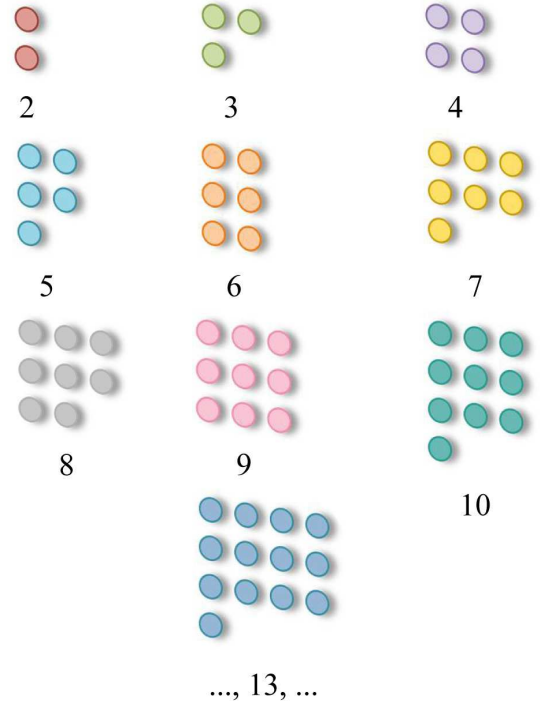


Fig. 2. The initial sampled coordinates for arbitrary convolutional kernel sizes are generated by an generation algorithm. It provides initial sampling shapes for irregular convolution kernel sizes.

tract features corresponding to irregularly sampled shapes. It is only necessary to reshape features and use the corresponding convolution operation. So in Fig. 3, the final "Reshape" and "Conv" represent any of the above methods. Moreover, to clearly show the process of AKConv, after resampling in Fig. 3, we put the dimension of the feature corresponding to the convolution in the third dimension. However, When the code is implemented, it is located in the last dimension.

Following RFACnv and Deformable Conv, we stack the resampled features in the column direction and then use row convolution with size $(N, 1)$ and step size $(N, 1)$. Therefore, AKConv can perfectly accomplish the irregular convolutional feature extraction process. AKConv completes the process of feature extraction by irregular convolution, and it can flexibly adjust the sample shape according to the offset and bring more exploration options for convolutional sampling shapes. Unlike Standard Convolution and Deformable Conv, they are limited by the idea of a regular convolution kernel.

3.3 Extended AKConv

We consider the design of AKConv to be a novel design that accomplishes the feat of extracting features from irregular and arbitrarily sampled shape convolutional kernels. Even without using the offset idea in Deformable Conv, AKConv can still make a variety of convolution kernel shapes. Because, AKConv can resample with the initial coordinates to present a variety of changes. As shown in Fig. 4, we design various initial sampling shapes for convolution of size 5. In Fig. 4, we only show some examples of size 5. However, the size of AKConv can be arbitrary, therefore as the size increases, the initial convolutional sampling shapes of AKConv become richer and even infinite. Given that the target shape varies across datasets, it is crucial to design the convolution operation corresponding to the sampled shape. AKConv fully realizes it

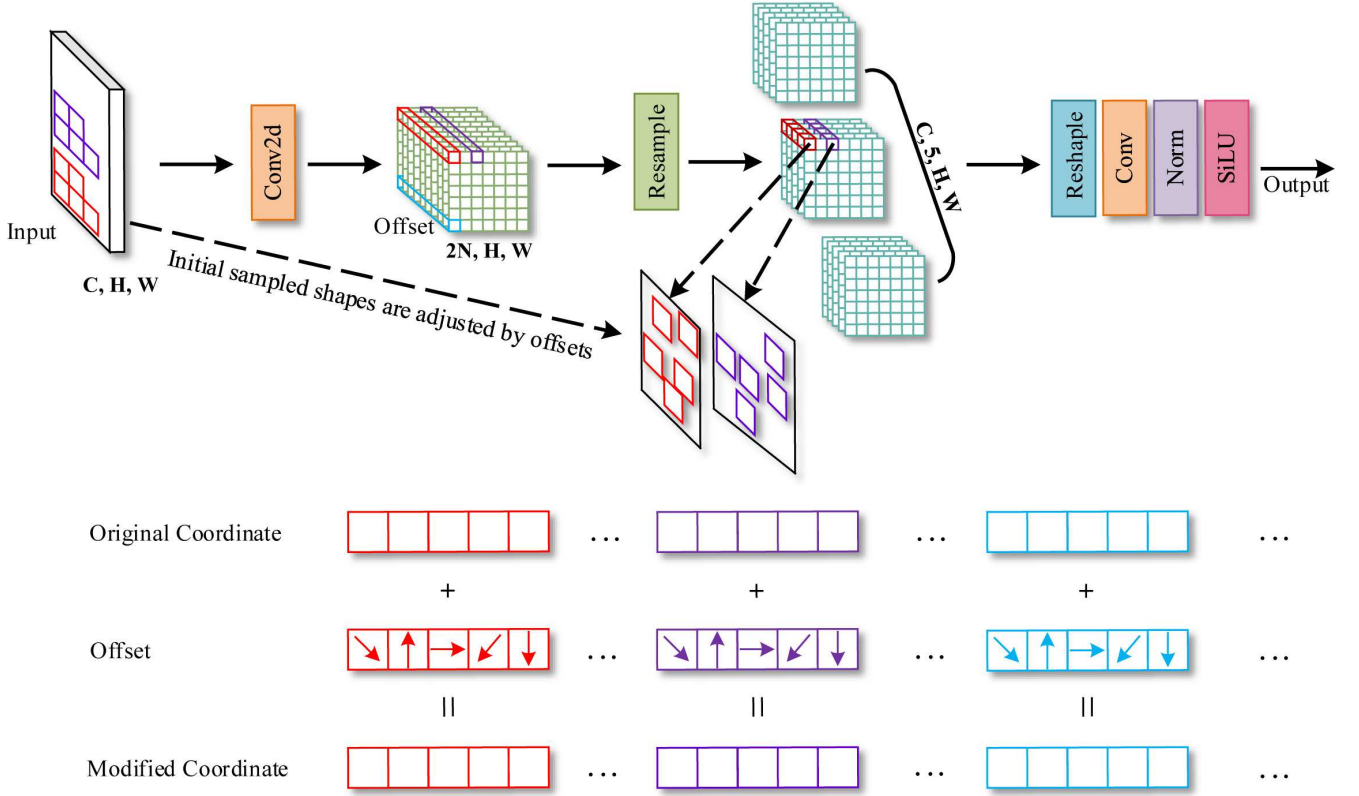


Fig. 3. It shows a detailed schematic of the structure of AKConv. It assigns initial sampling coordinates to a convolution of arbitrary size and adjusts the sample shape with the learnable offsets. Compared to the original sample shape, the sample shape at each position is changed by resampling.

by designing the convolution operation with the corresponding shape according to the phase-specific domain. It can also be similar to Deformable Conv by adding a learnable offset to dynamically adapt to changes of the object. For a specific task, the design of the initial sampling location of the convolution kernel is an important, because it is an a prior knowledge. As in Qi et al. [27], they proposed sampling coordinates with corresponding shapes for the elongated tubular structure segmentation task, but their shape selection was only for elongated tubular structures.

AKConv really achieves the process of convolution kernel operation with any number of arbitrary shapes, and it can make the convolution kernel present a variety of shapes. Deformable Conv [11] was designed to compensate for the shortcomings of regular convolution. Whereas DSConv [27] was designed for specific object shapes. They have not explored convolution of arbitrary size and convolution of arbitrary sample shapes. The design of AKConv remedies these problems by allowing the convolution operation to efficiently extract the features of irregular sample shapes through Offset. AKConv allows the convolution to have any number of convolution parameters, and allows the convolution to take on a wide variety of shapes.

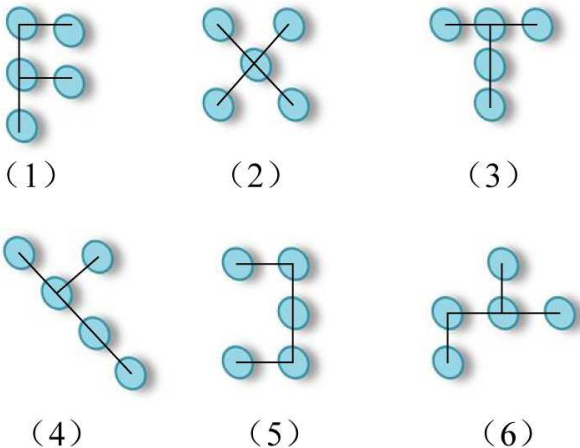


Fig. 4. It shows the initial sample shape of size 5. AKConv can achieve arbitrary sampling shapes by designing different initial sampling shapes.

4 Experiments

To verify the advantages of AKConv, we conduct rich target detection experiments based on advanced YOLOv5 [35], YOLOv7 [36] and YOLOv8 [19] respectively. All models in the experiments are trained based on RTX3090. To validate the advantages of AKConv we perform related experiments on representative COCO2017, VOC 7 + 12 and VisDrone-DET2021 datasets respectively.

4.1 Object detection experiments on COCO2017

COCO2017 includes train (118287 images), val (5000 images), and covers 80 object classes. It has become a standard dataset in the field of computer vision research, especially in the field of target detection. We choose the state-of-the-art YOLOv5n and YOLOv5s detectors as the baseline model. Then, AKConv with different sizes is used to replace the convolution

Table 1. Object detection AP_{50} , AP_{75} , AP , AP_S , AP_M , and AP_L on the COCO2017 validation sets. We adopt the YOLOv5n and YOLOv5s detection framework and replace the original convolution with the different size AKConv.

Models	AKConv	$AP_{50}(\%)$	$AP_{75}(\%)$	$AP(\%)$	$AP_S(\%)$	$AP_M(\%)$	$AP_L(\%)$	GFLOPS	Params(M)
YOLOv5n (Baseline)	-	45.6	28.9	27.5	13.5	31.5	35.9	4.5	1.87
	3	47.8	31.1	29.8	14.5	33.2	41	3.8	1.51
	5	48.8	32.6	31	14.6	34.1	43.2	4.1	1.65
	9	50.5	33.9	32.3	14.9	36.1	44.1	4.8	1.94
	13	51.2	34.5	33	15.7	36.3	45.6	5.5	2.23
YOLOv5s (Baseline)	-	57	39.9	37.1	20.9	42.4	47.8	16.4	7.23
	4	58.2	41.9	39.2	21.4	43.2	53.4	14.1	6.01
YOLOv5s	6	59.2	42.6	39.9	21.5	44.2	54.7	15.3	6.55
	7	59.4	43.2	40.4	21.5	44.6	55.1	15.9	6.82

Table 2. Based on the baseline dataset VOC 7 + 12, it is shown that AKConv can improves the mAP50 and mAP for YOLOv7-tiny.

Models	AKConv	Precision(%)	Recall(%)	mAP50(%)	mAP(%)	GFLOPS	Params(M)
YOLOv7-tiny (Baseline)	-	77.3	69.8	76.4	50.2	13.2	6.06
	3	80.1	68.4	76.1	50.3	12.1	5.56
	4	78.2	70.3	76.2	50.7	12.4	5.66
	5	77	71.1	76.5	50.8	12.6	5.75
YOLOv7-tiny	6	79.6	69.9	76.9	51	12.9	5.85
	8	78.6	70.1	76.7	51.2	13.4	6.04
	9	81	69.3	76.7	51.3	13.7	6.14

operations of YOLOv5n and YOLOv5s. The replacement details are the same as the target detection experiments in [28]. In the experiments, the default parameters of the network are used except for the epoch and batch-size parameters. Based on a batch size of 32, we trained each model for 300 epochs. Following previous work, we report AP_{50} , AP_{75} , AP , AP_S , AP_M and AP_L . Moreover, we also report target detection on YOLOv5n and YOLOv5s for AKConv with sizes 5, 4, 6, 7, 9, and 13, respectively. As shown in Table 1, the detection accuracy of YOLOv5 gradually increases with the increase of the convolutional kernel size, while the number of parameters required by the model and the computational overhead also gradually increase. Compared to standard convolutional operations, AKConv substantially improves the target detection performance of YOLOv5 on COCO2017. It can be seen that when the size of AKConv is 5, it not only makes the number of parameters and computational overhead required by the model decrease, but also significantly improves the detection accuracy of YOLOv5n. Its AP_{50} , AP_{75} , and also AP are all improved by three percentage points, which is outstanding. AKConv improves the AP_S , AP_M , and AP_L of the baseline model, but it is obvious that AKConv improves the detection accuracy of large objects significantly compared to small and middle objects. We assert that AKConv uses offsets to better adapt to the shape of large objects.

4.2 Object detection experiments on VOC 7+12

In order to further validate our method, we conduct experiments on the VOC 7+12 dataset, which is a combination of VOC2007 and VOC2012, comprising 16551 training sets and 4952 validation sets, and covers 20 object categories. To

test the generalizability of AKConv across different architectures, we selected YOLOv7-tiny as the baseline model. Since YOLOv7 and YOLOv5 are systems with different architectures, it is possible to compare the performance of AKConv with different architectural settings. In YOLOv7-tiny, we use AKConv with different sizes to replace standard convolutional operation. The details of the replacement follows the work in [28]. The hyperparameter settings for all models are consistent with those in the previous section. Following previous work, we present both mAP50 and mAP. As demonstrated in Table 2, with the increasement of size in AKConv, the network’s detection accuracy gradually improves, while the model’s parameter count and computational demand also incrementally rise. These experiments further substantiate the advantages of AKConv.

4.3 Object detection experiments on VisDrone-DET2021

In order to verify again that AKConv has strong generalization ability, based on VisDrone-DET2021 data, we conducted relevant target detection experiments. VisDrone-DET2021 is a challenging dataset taken by UAVs in different environments, weather and lighting conditions. It is one of the largest datasets with the widest coverage of UAV aerial photography in China. The number of training sets is 6471 and the number of validation sets is 548. As in Section 4.1, we choose YOLOv5n as the baseline to use AKConv to replace convolutional operations in the network. In experiments, the batch-size is set 16 to facilitate the exploration of larger convolution sizes, and all other hyperparameter settings are the same as before. As in the previous section, we report mAP50 and mAP, respectively. As shown in Table 3, it is clear to see that

Table 3. Object detection mAP50 and mAP on the VisDrone-DET2021 validation set by using different size of AKConv to replace convolutional operation.

Models	AKConv	Precision(%)	Recall(%)	mAP50(%)	mAP(%)	GFLOPS	Params(M)
YOLOv5n (Baseline)	-	38.5	28	26.4	13.4	4.2	1.77
	3	37.9	27.4	25.9	13.2	3.5	1.41
	5	40	28	26.9	13.7	3.8	1.56
	6	38.1	28.1	26.8	13.6	4	1.63
YOLOv5n	7	39.8	28.2	27.5	14.2	4.2	1.7
	9	39.7	28.9	27.7	14.3	4.5	1.84
	11	40.4	28.8	27.7	14.2	4.8	1.99
	14	40	28.8	27.9	14.3	5.3	2.2

Table 4. Object detection AP_{50} , AP_{75} , AP , AP_S , AP_M , and AP_L on the COCO2017 validation sets. We compare the performance of the AKConv, Deformable Conv and DSConv with same size.

Models	AP_{50} (%)	AP_{75} (%)	AP (%)	AP_S (%)	AP_M (%)	AP_L (%)	GFLOPS	Params(M)
YOLOv5s	54.8	37.5	35	19.2	40	45.2	16.4	7.23
YOLOv5s (DSConv=5)	43.2	23.5	23.9	13	27.6	30.5	14.8	6.45
YOLOv5s (AKConv=5)	56.6	40.7	38	20.8	41.8	52	14.8	6.45
YOLOv5s (AKConv=9)	57.8	41.4	38.7	20.8	42.8	52.3	17.1	7.37
YOLOv5s (AKConv=9, Padding=1)	58.3	41.9	39.2	21.6	43.2	53.5	17.1	7.37
YOLOv5s (Deformable Conv=3)	58.5	41.8	39.1	20.8	43.4	53.6	17.1	7.37
YOLOv5s (AKConv=11)	58.5	42.1	39.3	21.9	43.3	53.8	18.3	7.91
YOLOv5s (AKConv=11, Padding=1)	58.6	42.1	39.5	21.3	43.7	53.2	18.3	7.91

AKConv based on different sizes can be used as a lightweight option to reduce the number of parameters and computational overhead and improve network performance. In experiments, when the size of AKConv is set to 3, the detection performance of the model decreases compared to the baseline model, but the corresponding number of parameters and computational overhead are much smaller. Moreover, we can gradually adjust the size of AKConv to explore the changes in network performance. AKConv brings richer options to the network.

4.4 Comparison experiments

Unlike Deformable Conv [11], AKConv offers a richer choice for networks. AKConv compensates for the shortcomings of Deformable Conv, which only uses regular convolution operations, while AKConv can use both regular and irregular convolution operations. When the size of AKConv is set to the square of K , AKConv becomes a deformable Conv. Moreover, DSConv [27] also uses offsets to adjust the sampling shapes, but its sampling shape is designed for tubular targets, and the change of the sampling shape is limited. To contrast the advantages of AKConv, Deformable Conv, and DSConv at the same size. We perform experiments in COCO2017 and VOC 7 + 12 based on YOLOv5s and YOLOv5n. As shown in the Table 4 and Table 5. When the number of convolution kernel parameters is 9 (i. e., the standard 3×3 convolution), it can be seen that the performance of AKConv and Deformable Conv is the same. Because when the convolution kernel size is regular, the AKConv is the Deformable Conv. But we have mentioned that Deformable Conv has not explored the irregular convolution kernel size. Therefore, a convolution oper-

ation with a number of parameters of 5 and 11 cannot be implemented. When designing AKConv, we not implement zero-padding for input features. However, in Deformable Conv padding is used. Therefore, for a fair comparison, in AKConv, we also utilize zero-padding for input features. Experiments show that zero-padding in AKConv helps the network to improve performance. Since DSConv is designed for a specific tubular shape, it can be seen that its detection performance on COCO2017 and VOC 7 + 12 is not obvious. When implementing DSConv, Qi et al. [27] expands the features of rows or columns, and finally used column convolution or columns convolution to extract features similar to us. So their method can also implement convolution operations with parameters 2, 3, 4, 5, 6, 7, etc. Under the same size, we also conduct a comparison experiment. Because, the DSConv not completes the down-sample method, in experiments, we use the AKConv and DSConv to replace 3×3 convolution in C3 for YOLOv5n. Experimental results are shown in Table 4 and Table 5. AKConv is advantageous over DSConv, because DSConv is not designed to improve the performance of convolutional kernels of arbitrary size, but rather to explore for targets of specific shapes. In contrast, AKConv provides a rich choice of convolutional kernel selection and exploration that can effectively improve network performance.

4.5 Exploring the initial sampled shape

As mentioned earlier, AKConv can extract features by using arbitrary sizes and arbitrary sample shapes. To explore the effect of AKConv with different initial sample shapes on the network, we conducted experiments at COCO2017 and VisDrone-DET2021, respectively. On COCO2017, we con-

Table 5. Based on VOC 7 + 12, we compared other sizes of AKConv and DSConv and reported detection accuracy and other evaluation metrics, respectively.

Models	Precision(%)	Recall(%)	mAP50(%)	mAP(%)	GFLOPS	Params(M)
YOLOv5n	73.8	62.2	68.1	41.5	4.2	1.77
YOLOv5n (DSConv=4)	63	50.4	54.2	26.1	3.7	1.55
YOLOv5n (AKConv=4)	76.5	63.6	70.8	46.5	3.7	1.55
YOLOv5n (DSConv=9)	60.6	50.8	53.4	25.3	4.8	1.9
YOLOv5n (AKConv=9)	76.7	65.2	71.8	48.4	4.8	1.9

Table 6. Based on COCO2017 and YOLOv8n, we explore the different size of AKConv with different initial sampled shapes. The "Sampled Shape i" denotes different initial sampled shapes of AKConv.

Models	$AP_{50}(\%)$	$AP_{75}(\%)$	AP	$AP_S(\%)$	$AP_M(\%)$	$AP_L(\%)$	GFLOPS	Params(M)
YOLOv8n	49	37.1	34.2	16.9	37.1	49.1	8.7	3.15
YOLOv8n-5 (Sampled Shape 1)	49.5	37.6	34.9	16.8	38.2	50.2	8.4	2.94
YOLOv8n-5 (Sampled Shape 2)	49.6	37.8	34.9	15.9	38.4	50.1	8.4	2.94
YOLOv8n-5 (Sampled Shape 3)	49.6	38.1	35	16.6	38.2	50.9	8.4	2.94
YOLOv8n-6 (Sampled Shape 1)	50.1	38.3	35.3	16.6	38.6	51.1	8.6	3.01
YOLOv8n-6 (Sampled Shape 2)	50.2	38.2	35.4	16.6	38.3	51.3	8.6	3.01

Table 7. It is shown that different initial sampled shapes of AKConv obtain the performance of YOLOv5n on VisDrone-DET2021.

Models	Shapes	Precision(%)	Recall(%)	mAP50(%)	mAP(%)
YOLOv5n	a	39.5	27.9	26.9	13.7
	b	39.4	28.2	26.8	13.6
	c	37.4	27.8	26.1	13.4
	d	37.5	27	25.5	12.9
	e	38.4	27.6	26.4	13.4

ducted experiments based on a batch-size of 32 and an epoch of 100. In VisDrone-DET2021, we conducted experiments based on a batch-size of 16 and an epoch of 300. All other hyperparameters are network defaults. In COCO2017, we choose YOLOv8n for our experiments. As shown in Table 6, AKConv can still improve the detection accuracy of the network. The network structures of YOLOv8 and YOLOv5 are similar. The difference is the design of C3 and C2f. It can be seen that the performance increase obtained by adding AKConv in YOLOv8 is not as good as in YOLOv5. We think that YOLOv8 needs more parameters than YOLOv5 under the same size, so more number of parameters can provide better feature information as AKConv does. Therefore with the addition of AKConv, the YOLOv8 boost is not as significant as the YOLOv5. Furthermore, at the same size, we test the effect of different initial sample shapes on network performance in COCO2017. It is obvious that under different initial samples, the fluctuation of the detection accuracy obtained by the network is not large. It benefits from the fact that the massive data of COCO2017 can flexibly adjust the offset. But, it does not mean that the network obtains detection accuracy that are not significantly different at all initial sampling coordinates. To explore again the effect of AKConv with different initial shapes on the network, we explore AKConv with size 5 and with different initial samples for experiments based

on YOLOv5n on VisDrone-DET2021. It can be seen in Table 7 that the network obtains different detection accuracy with different initial samples. Therefore, AKConv with different initial sampling shapes has an impact on the performance of the network. Moreover, for specific networks and datasets, it is important to explore AKConv with appropriate initial sampling shapes to improve network performance.

5 Analysis and discussions

We initially AKConv of size 5 at different sampling positions in the previous experiment to observe the detection performance of YOLOv5n. It can be clearly noticed that the network behaves differently under different initial sampling shapes. It suggests that the adjustment ability of offsets is also limited. To measure the change in offset at each given position, we give the definition of the Average Offset, which is defined as follows:

$$AO = (\sum_i^{2N} |Offset_i|) / (2N) \quad (3)$$

AO (Average Offset) measures an average degree of change in the sampled points at each position by summing the offsets, and then taking the average. To observe the change of offsets, we selected the trained network and choose the last layer of AKConv to analyze the overall change trend of offsets. For the analysis, we randomly selected four images in VisDrone-DET2021 and then visualized the AKConv of size 5, which is initial for different sampling positions. As shown in Fig. 5, we visualized the degree of change AO of offset at each sampling location. The different colors in Fig. 5 represent the change in offsets at each sample position for different initial samples after training. The color of the line corresponds to the initial sampling shape in the middle. The different initial sample shapes in Fig. 5 correspond to the initial sample shapes in Table 7. It can be concluded that OA changes less for the blue and red initial sample shapes in Fig. 5. It means

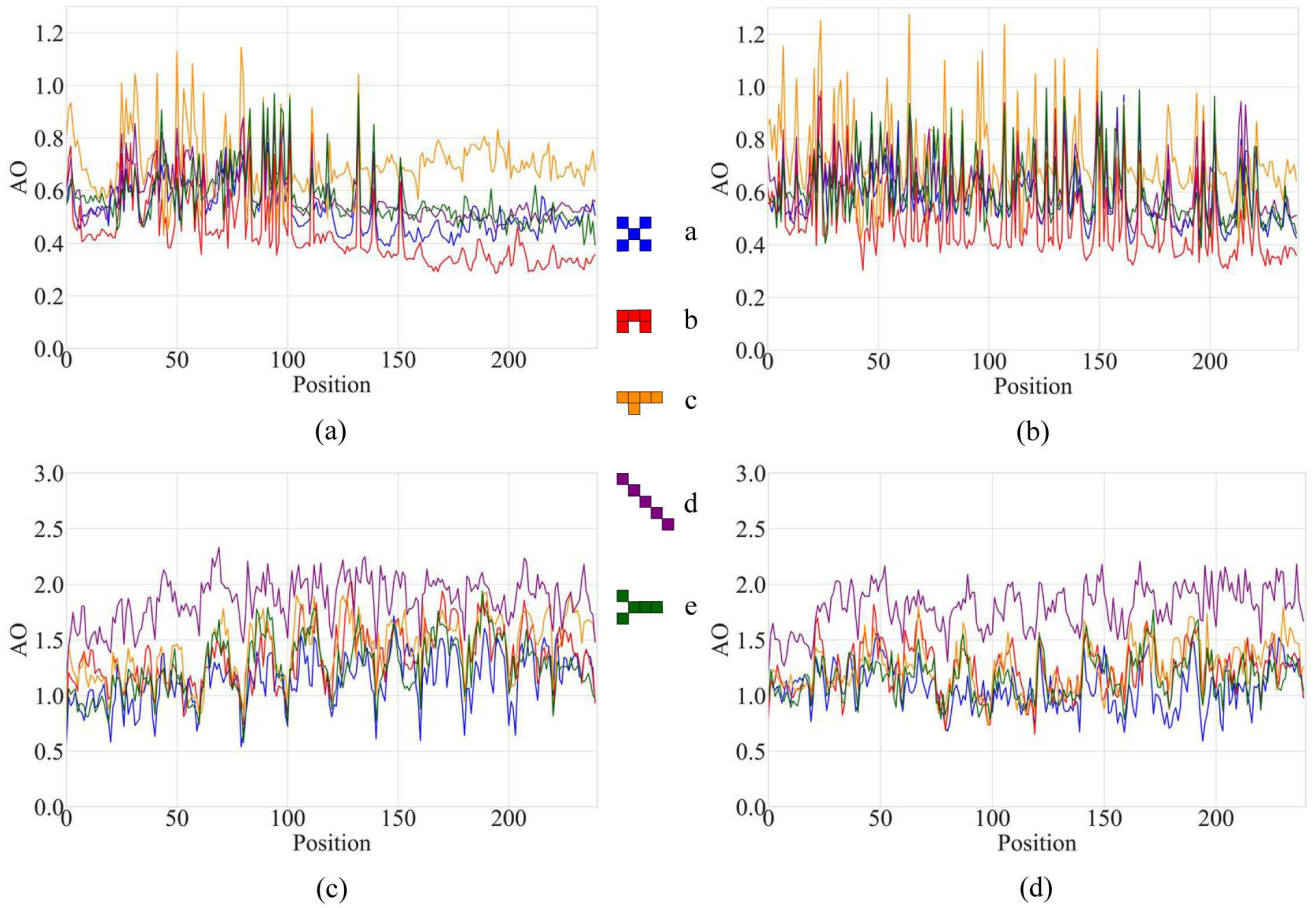


Fig. 5. It shows the variation of AO of AKConv for different initial sample shapes of size 5. It can achieve arbitrary sampling shapes by designing different initial sampling shapes for AKConv.

the red and blue initial samples are more suitable for this dataset than the other initial samples. As in the experiment in Table 7, it can be seen that the initial sampling shapes corresponding to blue and red obtained better detection accuracy. All the experiments proved that AKConv is able to bring significant performance improvement to the network. Unlike Deformable Conv, AKConv has the flexibility to scale network performance based on size. In all the experiments, we explore AKConv with size 5 extensively. Because when training COCO2017 with a large amount of data, we found that when setting the size of AKConv to 5, the training speed is not much different from the original model. Moreover, as the size of AKConv increases, the training time gradually increases. In the experiments of COCO2017, VOC 7+12, and VisDrone-DET2021, AKConv with size set to 5 gave good results for the network. Of course, the exploration of AKConv for other sizes is possible because the number of parameters that show linear growth and arbitrary sampling shapes bring a wealth of choices for the exploration of AKConv. AKConv can realize convolution operation with arbitrary size and arbitrary samples, and can automatically adjust the sample shape to adapt to the target change by offsets. All experiments demonstrate that AKConv improves network performance and provides richer options for the trade-off between network overhead and performance.

6 Conclusion

It is obvious that in real life as well as in the field of computer vision, the shapes of objects show various variations. The

fixed sample shape of convolutional operation cannot adapt to such changes. Although Deformable Conv can flexibly change the sample shape of convolution with the adjustment of offset, it still has limitations. Therefore, we propose AKConv, which truly realizes to allow convolution to have arbitrary sample shapes and sizes, which provides diversity in the choice of convolution kernels. Moreover, for different domains, we can design specific initial shapes of sampling coordinates to meet the real needs. Although in this paper, we have designed multiple shapes of sampling coordinates only for AKConv of size 5. However, the flexibility of AKConv is that it can target any size of sampling kernel to extract information. Therefore, in the future, we would like to explore AKConv with appropriate sizes and sample shapes for specific tasks in the field, which will add momentum to the subsequent tasks.

Bibliography

- [1] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778. [1](#)
- [2] G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4700–4708. [1](#)
- [3] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 779–788. [1](#)
- [4] C.-M. Chang, Y.-D. Liou, Y.-C. Huang, S.-E. Shen, P. Yu, T. Chuang, S.-J. Chiou, Yolo based deep learning on needle-type dashboard recognition for autopilot maneuvering system, *Measurement and Control* 55 (7-8) (2022) 567–582. [1](#)
- [5] Y. Xie, J. Zhang, C. Shen, Y. Xia, Cotr: Efficiently bridging cnn and transformer for 3d medical image segmentation, in: Medical Image Computing and Computer Assisted Intervention–MICCAI 2021: 24th International Conference, Strasbourg, France, September 27–October 1, 2021, Proceedings, Part III 24, Springer, 2021, pp. 171–180. [1](#)
- [6] M. Abbasi, A. Shahraki, A. Taherkordi, Deep learning for network traffic monitoring and analysis (ntma): A survey, *Computer Communications* 170 (2021) 19–41. [1](#)
- [7] H.-w. An, N. Moon, Design of recommendation system for tourist spot using sentiment analysis based on cnn-lstm, *Journal of Ambient Intelligence and Humanized Computing* (2022) 1–11. [1](#)
- [8] J. Qin, W. Pan, X. Xiang, Y. Tan, G. Hou, A biological image classification method based on improved cnn, *Ecological Informatics* 58 (2020) 101093. [1](#)
- [9] X. Wang, N. He, C. Hong, Q. Wang, M. Chen, Improved yolox-x based uav aerial photography object detection algorithm, *Image and Vision Computing* 135 (2023) 104697. [1](#)
- [10] E. Yang, W. Zhou, X. Qian, J. Lei, L. Yu, Drnet: Dual-stage refinement network with boundary inference for rgb-d semantic segmentation of indoor scenes, *Engineering Applications of Artificial Intelligence* 125 (2023) 106729. [1](#)
- [11] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, Y. Wei, Deformable convolutional networks, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 764–773. [1](#), [2](#), [3](#), [4](#), [6](#)
- [12] X. Zhu, H. Hu, S. Lin, J. Dai, Deformable convnets v2: More deformable, better results, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 9308–9316. [1](#)
- [13] Y. Zhao, L. Zhao, Z. Liu, D. Hu, G. Kuang, L. Liu, Attentional feature refinement and alignment network for aircraft detection in sar imagery, *arXiv preprint arXiv:2201.07124* (2022). [1](#)
- [14] T. Song, X. Zhang, D. Yang, Y. Ye, C. Liu, J. Zhou, Y. Song, Lightweight detection network based on receptive-field feature enhancement convolution and three dimensions attention for images captured by uavs, *Image and Vision Computing* (2023) 104855. [1](#)
- [15] S. Huang, Z. Lu, R. Cheng, C. He, Fapn: Feature-aligned pyramid network for dense image prediction, in: Proceedings of the IEEE/CVF international conference on computer vision, 2021, pp. 864–873. [1](#)
- [16] S. Zhao, S. Zhang, J. Lu, H. Wang, Y. Feng, C. Shi, D. Li, R. Zhao, A lightweight dead fish detection method based on deformable convolution and yolov4, *Computers and Electronics in Agriculture* 198 (2022) 107098. [1](#)
- [17] A. Bochkovskiy, C.-Y. Wang, H.-Y. M. Liao, Yolov4: Optimal speed and accuracy of object detection, *arXiv preprint arXiv:2004.10934* (2020). [1](#)
- [18] W. Yang, J. Wu, J. Zhang, K. Gao, R. Du, Z. Wu, E. Firkat, D. Li, Deformable convolution and coordinate attention for fast cattle detection, *Computers and Electronics in Agriculture* 211 (2023) 108006. [1](#)
- [19] J. Glenn, Ultralytics yolov8, <https://github.com/ultralytics/ultralytics> (2023). [1](#), [4](#)
- [20] D. Li, Y. Li, H. Sun, L. Yu, Deep image compression based on multi-scale deformable convolution, *Journal of Visual Communication and Image Representation* 87 (2022) 103573. [1](#)
- [21] T. Dumas, A. Roumy, C. Guillemot, Context-adaptive neural network-based prediction for image compression, *IEEE Transactions on Image Processing* 29 (2019) 679–693. [1](#)
- [22] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, N. Johnston, Variational image compression with a scale hyperprior, *arXiv preprint arXiv:1802.01436* (2018). [1](#)
- [23] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, A. Zisserman, The pascal visual object classes challenge: A retrospective, *International journal of computer vision* 111 (2015) 98–136. [2](#)
- [24] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C. L. Zitnick, Microsoft coco: Common objects in context, in: Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part V 13, Springer, 2014, pp. 740–755. [2](#)
- [25] P. Zhu, L. Wen, D. Du, X. Bian, H. Fan, Q. Hu, H. Ling, Detection and tracking meet drones challenge, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44 (11) (2021) 7380–7399. [2](#)
- [26] D. Li, J. Hu, C. Wang, X. Li, Q. She, L. Zhu, T. Zhang, Q. Chen, Involution: Inverting the inheritance of convolution for visual recognition, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 12321–12330. [2](#)
- [27] Y. Qi, Y. He, X. Qi, Y. Zhang, G. Yang, Dynamic snake convolution based on topological geometric constraints for tubular structure segmentation, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 6070–6079. [2](#), [4](#), [6](#)
- [28] X. Zhang, C. Liu, D. Yang, T. Song, Y. Ye, K. Li, Y. Song, Rfaconv: Innovating spatital attention and standard convolutional operation, *arXiv preprint arXiv:2304.03198* (2023). [2](#), [3](#), [5](#)
- [29] S. Woo, J. Park, J.-Y. Lee, I. S. Kweon, Cbam: Convolutional block attention module, in: Proceedings of the

- European conference on computer vision (ECCV), 2018, pp. 3–19. 2
- [30] Q. Hou, D. Zhou, J. Feng, Coordinate attention for efficient mobile network design, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2021, pp. 13713–13722. 2
- [31] Y. Chen, X. Dai, M. Liu, D. Chen, L. Yuan, Z. Liu, Dynamic convolution: Attention over convolution kernels, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 11030–11039. 2
- [32] M. Tan, Q. V. Le, Mixconv: Mixed depthwise convolutional kernels, arXiv preprint arXiv:1907.09595 (2019). 2
- [33] Q. Zhao, C. Zhu, F. Dai, Y. Ma, G. Jin, Y. Zhang, Distortion-aware cnns for spherical images., in: IJCAI, 2018, pp. 1198–1204. 2
- [34] B. Coors, A. P. Condurache, A. Geiger, Spherenet: Learning spherical representations for detection and classification in omnidirectional images, in: Proceedings of the European conference on computer vision (ECCV), 2018, pp. 518–533. 2
- [35] J. Glenn, YOLOv5 release v6.1, <https://github.com/ultralytics/yolov5/releases/tag/v6.1> (2022). 4
- [36] C.-Y. Wang, A. Bochkovskiy, H.-Y. M. Liao, YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 7464–7475. 4