sciendo

JAISCR, 2025, Vol. 15, No. 2, pp. 197 – 210
10.2478/jaiscr-2025-0010

ADAPTIVE SEPARATION FUSION: A NOVEL DOWNSAMPLING APPROACH IN CNNS

Xia Ji[1,2,*], Jinglong Chang[1], Yapeng Ji[1]

[1]*School of Computer Science and Technology, Anhui University*

[2]*Anhui Provincial International Joint Research Center for Advanced Technology in Medical Imaging*

[*]*E-mail: jixia1983@163.com*

Submitted: 13th October 2024; Accepted: 14th January 2025

Abstract

Almost all computer vision tasks rely on convolutional neural networks and transformers, both of which require extensive computations. With the increasingly large size of images, it becomes challenging to input these images directly. Therefore, in typical cases, we downsample the images to a reasonable size before proceeding with subsequent tasks. However, the downsampling process inevitably discards some fine-grained information, leading to network performance degradation. Existing methods, such as strided convolution and various pooling techniques, struggle to address this issue effectively. To overcome this limitation, we propose a generalized downsampling module, Adaptive Separation Fusion Downsampling (ASFD). ASFD adaptively captures intra- and inter-region attentional relationships and preserves feature representations lost during downsampling through fusion. We validate ASFD on representative computer vision tasks, including object detection and image classification. Specifically, we incorporated ASFD into the YOLOv7 object detection model and several classification models. Experiments demonstrate that the modified YOLOv7 architecture surpasses state-of-the-art models in object detection, particularly excelling in small object detection. Additionally, our method outperforms commonly used downsampling techniques in classification tasks. Furthermore, ASFD functions as a plug-and-play module compatible with various network architectures.

Keywords: adaptive, object detection, image classification, deep learing, downsampling

1 Introduction

In recent years, the continuous advancement of deep learning has driven the development of various network models in computer vision, such as VGG [1], ResNet [2], R-CNN [3], UNet [4], SINextNet [5], and DETR [6]. These models extract image features through multiple convolutions. However, performing convolutions on images at their original resolution consumes significant computational resources. To address this, deep learning workflows typically downsample feature maps multiple times to reduce their dimensionality before further processing. Despite its efficiency, downsampling exponentially reduces the size of feature maps, leading to the loss of crucial texture information. After multiple downsampling operations, this accumulated information loss becomes significant, adversely af-

fecting inference results and reducing accuracy in various visual tasks. Therefore, an effective downsampling method should not only minimize information loss but also learn the relationships between spatial and channel dimensions effectively.

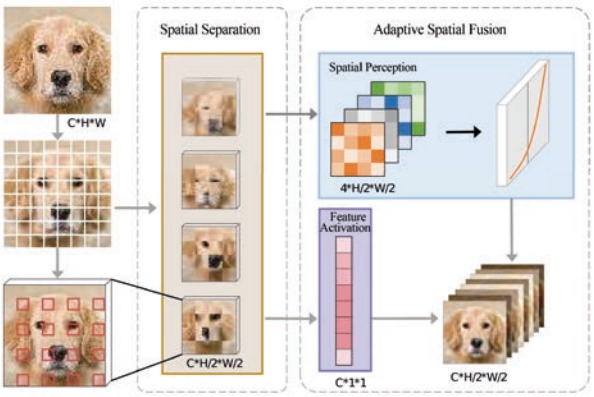


Figure 1. This figure shows ASFD structure when the downsampling scaling factor is 2. There are two stages, one for spatial separation and one for adaptive spatial fusion. Adaptive spatial fusion contains two branches, the spatial perception and the feature activation. The white block in the spatial perception is the Softmax function on the channel.

Currently, neural networks typically use pooling and strided convolution for downsampling. Pooling reduces the size of feature maps and is translation invariant, as it operates on local image regions. This ensures that similar feature representations can be extracted even when the input image is shifted. Early pooling methods, such as Max Pooling and Average Pooling [7]. Boureau et al. [8] investigated the robustness and usability of these methods and found that Max Pooling produces representative outputs with low feature activation. However, pooling methods have inherent limitations: Max Pooling retains only the maximum value in a local region, discarding other information and potentially losing details, while Average Pooling computes the average, which can blur subtle features. Subsequent researchers have proposed several enhanced pooling algorithms [9–15]. These algorithms include random pooling, which utilizes weighted sampling of probabilities within each kernel, and S3Pool, which randomly samples rows and columns from the original feature map grid. While these methods mitigate some limitations, they do not completely resolve the inherent drawbacks of pooling. The second commonly used downsampling method is strided convolution, which extracts local features while reducing feature map dimensions by adjusting the stride of the convolution kernel. Unlike pooling, strided convolution involves more parameters, which can slow down the network and reduce its efficiency. In the early stages of a network, both pooling and strided convolution achieve effective downsampling, as high-resolution feature maps at these stages contain redundant information that offsets the loss caused by downsampling. However, as feature maps become smaller, both methods struggle to retain fine-grained details.

Recently, Zhu et al. [16] addressed this issue by adjusting feature sampling based on image context in the Fourier domain. While this approach alleviates the problem, it introduces additional computations, resulting in slower processing speeds. R et al. [17] proposed the SPD-Conv method, which takes a different approach. Instead of operating in the spatial domain, SPD-Conv transfers spatial information to the channel dimension. This preserves all input feature details and mitigates the loss of fine-grained information. However, in the later stages of the network, feature maps primarily represent abstract semantic information [18], where the dependency between neighboring grids plays a critical role in maintaining the integrity of this semantic information. Unfortunately, the SPD-Conv downsampling method separates neighboring grids, disrupting the spatial structure of feature maps and partially compromising the organization of semantic information.

To address this issue, we propose a novel downsampling method called Adaptive Separation Fusion Downsampling (ASFD) as shown in Fig. 1. Our approach consists of two stages: the spatial separation stage and the adaptive spatial fusion stage. In the spatial separation stage, grids are isolated within localized regions, while in the adaptive spatial fusion stage, grids are adaptively combined. The goal of the spatial separation stage is to preserve as much feature map information as possible, while the spatial fusion stage combines the previously separated grids smoothly using attentional relationships. This helps prevent damage to the spatial structure of the feature map. Specifically, in

the spatial separation part, our ASFD method divides the feature map into several sub-maps along the width and height dimensions. The grids that occupy the same spatial position within each sub-map are then arranged relative to their position in the original feature map. As a result, we obtain a set of intermediate feature maps after width-height scaling, all of which are achieved without introducing any additional parameters. In the adaptive spatial fusion part, two steps are involved: spatial perception and feature activation. In the spatial perception step, we measure the importance of pixels in the respective regions of the intermediate feature maps. Each pixel contributes to the initial downsampled feature map based on its importance. Then, in the feature activation step, attentional relationships between channels are identified. In this process, ASFD learns the importance of regions within each sub-map. This approach preserves all the information within the feature map, while enhancing the preservation of semantic information structure. In summary, this study makes the following contributions.

- We propose a novel plug-and-play downsampling module, ASFD, which efficiently learns image features while using fewer parameters.
- We evaluate the performance of ASFD in object detection, showing that the inclusion of the ASFD module reduces network parameters and significantly improves detection precision. Experimental results demonstrate that ASFD mitigates information loss during downsampling and enhances object detection performance.
- We compare the performance of ASFD with other downsampling methods in several typical network models. Our method consistently achieves higher classification accuracy, demonstrating that ASFD is a versatile plug-and-play module applicable to various networks.

2 Related Work

Deep learning models are neural networks with multiple hidden layers. As the number of layers increases, so do the parameters and computational complexity. When using the original-sized image as input, significant computational resources are required. Therefore, image processing in deep learning often involves downsampling the original image to reduce its size. However, downsampling inherently leads to a loss of feature information. Addressing this challenge is essential for improving the quality of learning data. Researchers must aim to minimize this loss while preserving the overall effectiveness of the deep learning network.

Downsampling, as an integral component of neural networks, significantly influences the performance of downstream tasks. In 2012, AlexNet [19] won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) competition, employing maximum pooling for downsampling. In 2014,the VGG model [1] from Oxford University, which also uses maximum pooling for downsampling, was introduced. In 2015, ResNet [2] claimed victory in the ILSVRC, utilizing a convolutional layer with a stride of 2 for downsampling. Unet [4], an outstanding semantic segmentation model, performs pixel-level classification and also employs maximum pooling for downsampling. Additionally, many transformer-based visual models, such as Vit [20], DETR [6], typically pass the input through a convolutional backbone, where downsampling is performed using strided convolution. Moreover, numerous later models have adopted convolution, pooling, or a combination of both for downsampling.

Convolution and pooling each have their distinct advantages. Convolution extracts information from the feature map, preserving key features during the downsampling process. However, it involves parameters and is relatively slow. Max Pooling, while effective for downsampling, may discard some fine-grained details, whereas Average Pooling tends to blur subtle features, making it less suitable for fine-grained feature processing [8]. Some methods have been proposed to enhance pooling techniques to better preserve relevant features during downsampling. Random pooling uses weighted sampling of probabilities in each kernel [10]. Mixed pooling randomly combines maximum and average pooling based on probabilities [11], and Power Average pooling learns a parameter p to determine the importance of maximum and average pooling [12]. S3Pool randomly samples rows and columns of the original feature map grid [13]. Local Importance Pooling utilizes learnable weights as

a subgrid-based attention mechanism [14]. Zhao et al. [15] proposed LiftPool which is based on four different learnable sub-bands to produce outputs. SoftPool is based on the SoftMax weighting method to maintain the basic attributes of the inputs while amplifying feature activations of greater intensity [9]. While these improved methods preserve more feature information to some extent, they do not fully address the inherent limitations of pooling. Fundamentally solving the problem of information loss remains a significant challenge.

Recently, Zhu et al. [16] proposed Fouridown, a method that learns the frequency of each location in a two-dimensional plane using Fourier functions. It parameterizes the static, parameter-free weighted downsampling operator as a learnable, context-adaptive operator within a uniform Fourier framework. This approach adaptively learns both spatial and channel frequencies, which helps alleviate information loss to some extent. However, due to its complex components, it introduces additional computational overhead. And the recently proposed SPD-Conv [17] divides the feature map into several equal-sized sub-blocks. It extracts pixels at corresponding positions within each sub-block, then rearranges them based on the spatial position of the sub-blocks. This results in feature maps with reduced width and height but unchanged channels. These feature maps are concatenated along the channel dimension, and the number of channels is further reduced using point-wise convolution. This method transfers spatial information to the channels while retaining all input feature information. However, in the later stages of a convolutional network, feature maps represent abstract semantic information and contain fewer pixels. Unfortunately, this method partially disrupts the spatial and semantic structure of the feature maps. To address this structural limitation, we propose spatial fusion for smooth downsampling. This approach preserves the spatial structure of the feature map while retaining essential feature information.

3 Adaptive Separation Fusion Downsampling

3.1 Symbol Definition and Theoretical Analysis

This section describes our proposed new downsampling module called ASFD. First, we introduce the symbols used in this paper and their corresponding meanings, as shown in Table 3.1.

Table 1. Symbol definition

Notation	Meanings
F	Feature map
$\mathbb{R}^{A\times B\times C}$	Three-dimensional space with dimensions A, B, and C
s	Downsampling scaling factor
F_{sub}	Sub-map within a feature map
$X(i,j)$	Sub-map with spatial position (i,j)
$A(i,j)$	Three-dimensional matrix where elements indexed i in the first dimension and j in the second dimension are 1
$F'_{x,y}$	The (x,y)-th intermediate feature map
F'_i	The intermediate feature maps
H_p	Spatial perception marker
H_f	Feature activation marker
F_{pi}	Fusion weights after convolution in spatial perception
w_i	MLP layer weights
H_p^i	Spatial perception attention map
F_{down}	The feature map obtained after downsampling

Downsampling definition: In a feature map $F(F\in R^{C\times H\times W})$ with scaling factor s(the common factor of W and H), the process of reducing the feature map to $F_{af}(F_{af}\in R^{C'\times\frac{H}{s}\times\frac{W}{s}})$ by a series of functional transformations D is called downsampling, where $s\geq 2$, and is an integer. When the scaling factor is 2, the structure of our ASFD module is shown in Fig. 1.

Neural networks typically employ strided convolution or pooling for downsampling. Strided convolution skips over certain information in the feature map. As the stride increases, the distance between regions modeled by the convolutional kernel grows, making it difficult to capture connections between closely spaced information [17]. This results in a direct loss of information. Similarly, pooling methods also introduce biases: max pooling tends to lose fine details by focusing only on dominant features, while average pooling can blur subtle features, both of which are detrimental to precise feature processing [8]. In contrast, the spatial separation stage of ASFD adopts a process of separation and merging, avoiding strided feature extraction

and thereby preserving all feature information. In the second stage, two branches are utilized to model feature relationships both within and between regions, adaptively learning the dependencies among the separated features and enabling smooth downsampling.

3.2 Spatial Separation

First, the feature map goes through a spatial separation stage. In the feature map $F(F \in R^{C\times H\times W})$ to be downsampled, C is the number of channels in the feature map, H represents the height, and W represents the width. The downsampling scaling factor is s. We divide F into $\frac{H}{s} \times \frac{W}{s}$ sub-maps along the W and H, with each sub-map $F_{sub} \in R^{C\times s\times s}$. We use $X(i,j)$ to denote the sub-map of the original feature map at the i-th row and j-th column along the channel dimension C, where $i,j \in \left(0, \frac{H}{s}-1\right)$. The grids occupied the same position in each F_{sub} are rearranged to a two-dimensional plane based on their respective (i,j) indices. There are $s\times s$ grids in F_{sub} in each channel, so s^2 intermediate feature maps are obtained after this process. Using $A_{(i,j)}$, we represent a three-dimensional matrix with size $C\times s\times s$. The element in this matrix at dimension 1 index i and dimension 2 index j is 1. $E_{m\times n}$ is a 3-dimensional matrix with size $C\times m\times n$, and all of its elements are 1. The s^2 intermediate feature maps obtained by the downsampling process are as follows:

$$F'_{x,y} = \sum_{i=0}^{\frac{H}{s}-1} \sum_{j=0}^{\frac{W}{s}-1} \left(E_{\frac{H}{s}\times s} \otimes (X(i,j) \circ A(x,y)) \otimes E_{s\times \frac{W}{s}} \circ A(i,j) \right)$$

where $x,y \in [0, s-1]$.

After the abovementioned steps, we obtain s^2 feature maps $F_i^{'}(F_i^{'} \in R^{C\times \frac{H}{s}\times \frac{W}{s}})$, where $i \in \left(1, s^2\right)$, and then sum them element-wise by position. After the aforementioned transformation, the pixels in the feature map simply result from the splicing of unsampled feature information without further learning of the spatial structure and semantic information of the features. However, if each sub-map is considered an independent feature map, it exhibits varying strengths and attention regions for features within its own space.

3.3 Adaptive Spatial Fusion

Therefore, we design the adaptive spatial fusion stage, which comprises two branches named spatial perception and feature activation. These branches are denoted by the transformation symbols H_p and H_f. The branch structure is illustrated in Fig. 2.

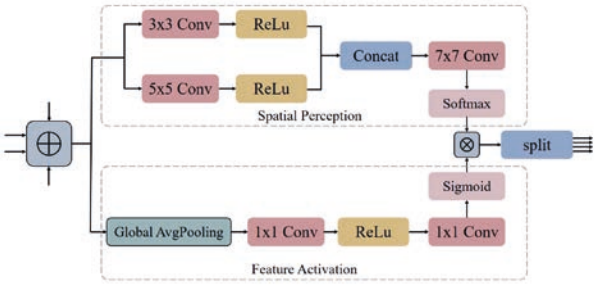


Figure 2. Adaptive Spatial Fusion Architecture.

Spatial perception branch: In the process of separating submaps, the relationship between neighboring submaps is inseparable and interdependent. This consideration is particularly relevant because some large objects often exceed the area covered by a single sub-map. Consequently, the spatial structure relationship between neighboring submaps must be accurately represented. In convolutional neural networks, employing kernels of varying sizes allows us to capture different scales of local information. These kernels effectively represent the feature spatial relationships within and between submaps. Therefore, we utilize distinct kernel sizes, which enables us to extract contextual information from feature maps at different scales. We use smaller 3×3, 5×5 kernels to extract localized features within a short range. The aim is to capture spatial fusion relationships between and within sub-map regions, which results in the generation of fusion weights denoted as F_{p1} and $F_{p2}(F_{p1}, F_{p2} \in R^{C\times \frac{H}{s}\times \frac{W}{s}})$. Subsequently, F_{p1}, F_{p2} are connected by the channel, and then they go through a convolution layer with 7×7 kernel, the larger receptive field can extract feature information at different scales and learn the connection between them. At this point we obtain the fusion weight $F_{p3}(F_{p3} \in R^{C\times \frac{H}{s}\times \frac{W}{s}})$. Finally, the SoftMax operation is applied along the channel dimensions of fusion weights F_{p3} to obtain the spatial perception attention map for each downsampled feature map. This attention map re-

flects the importance of each sub-map region, as suggested by the context. The spatial perception is then computed as follows:

$$\begin{aligned} H_p &= S\left(f_{7\times7}\left[f_{3\times3}\left(\sum F_i'\right); f_{5\times5}\left(\sum F_i'\right)\right]\right) \\ &= S(f_{7\times7}[F_{p1}; F_{p2}]) \\ &= S(F_{p3}) \end{aligned}$$

where S denotes SoftMax and $f^{*\times*}$ denotes a convolution layer with kernel *.

Feature activation branching: Globally, each channel of the feature map may not be equally meaningful for different objects. Therefore, the method of enhancing the features of important channels while attenuating those of channels that contribute relatively less is feasible and effective [21]. ASFD adaptively activates channels by sensing the importance of each channel feature. We compress the spatial dimension of the input feature map to efficiently compute feature activation levels. Previous works proposed that average pooling can efficiently learn the range of the object object. In addition, average pooling can capture strong and weak relationships among channels in different feature maps. Therefore, we exclusively use global average pooling to aggregate spatial information from the feature maps for speed and efficiency. The channel attention map $F_c(F_c \in R^{C\times1\times1})$ is obtained after averaging the pooled features; then, the feature activation map $H_f(H_f \in R^{C\times1\times1})$ is generated by the multilayer perceptron:

$$H_f = \sigma\left(W_1\left(W_0\left(\text{AvgPool}\left(\sum F_i'\right)\right)\right)\right)$$

where the MLP consists of a two-layer perceptron with transformation matrices denoted as $w_0 \in R^{\frac{C}{r}\times C}$, $w_1 \in R^{C\times\frac{C}{r}}$, and r is the channel reduction rate. σ denotes the Sigmoid, and ReLu activation function is applied after w_1.

After processing through two branches, we split H_p channel-wise and obtain s^2 spatial fusion attention maps H_i^p, where the channel is 1 and $i \in \left(1, s^2\right)$. We multiply them with H_f to obtain the adaptive fusion weight $W_i(W_i \in R^{s^2\times\frac{H}{s}\times\frac{W}{s}})$, where $i \in \left(1, s^2\right)$. They are then multiplied with the intermediate feature maps $F_i^{'}$ obtained from spatial separation stage. Finally, we sum them up element-wise by position to obtain the final downsampling result. In summary, the computational results of ASFD are as follows:

$$F_{\text{down}} = \sum_{i=1}^{s^2}\left(H_p^i \circ H_f \circ F_i'\right) = \sum_{i=1}^{s^2}\left(W_i \circ F_i'\right)$$

Finally, the number of channels was adjusted to get a downsampled map $F_{down}(F_{down} \in R^{C^{'}\times\frac{H}{s}\times\frac{W}{s}})$.

4 Experiment

The experiments in this paper first validate the performance of ASFD in object detection, supported by inference results and heat map visualizations. We then assess the performance of ASFD in image classification, followed by an investigation of the contribution of each component within ASFD. To demonstrate that ASFD effectively reduces information loss during downsampling, we use object detection as a case study, comparing the model's performance before and after integrating ASFD. We also present these effects in a more intuitive, visual format. Next, we verify the impact of ASFD on image classification across different networks, with experiments highlighting the versatility of our method. Finally, we conduct ablation studies on the spatial fusion and feature activation branches, showing that both branches are effective individually and work synergistically to enhance performance.

4.1 ASFD Performance in Object Detection

YOLOv7 is one of the more advanced, accurate, and fast single-stage detectors in object detection. Its downsampling layer employs a unique method that combines pooling and convolution, which effectively balances parameters, speed, and downsampling effects [22]. We use YOLOv7 as a baseline model to show the superiority of the ASFD more clearly.

First, we designed the YOLOv7-ASFD containing the ASFD module based on the characteristics of the YOLOv7. We replaced the downsampling layer of YOLOv7 with ASFD and changed the model structure accordingly. Our method significantly reduces the overall parameters compared

with the YOLOv7 baseline. We designed multiple variants, namely, YOLOv7-ASFD-tiny, YOLOv7-ASFD, and YOLOv7-ASFD-x, to evaluate ASFD's performance across different scales. Fig. 3 illustrates the structure of YOLOv7-ASFD.

To evaluate the effectiveness of ASFD in retaining essential information during downsampling and enhancing the model's performance, we trained and evaluated YOLOv7-ASFD on the NWPU VHR-10 dataset at various scales, comparing results with other detectors operating at similar scales. The NWPU VHR-10 dataset [23] comprises 650 positive images, each containing at least one detectable object, while the remaining 150 images do not contain any known objects. Among these images, 715 images are high-resolution remote sensing captures acquired from Google Maps, with a spatial resolution ranging from 0.5 m to 2 m. The remaining 85 images are pan-sharp color infrared images with an even higher spatial resolution of 0.08 m. In our experiments, we exclusively used positive images and split them into training and test sets following a 6:4 ratio. The validation set accounts for 20% of the overall training set.

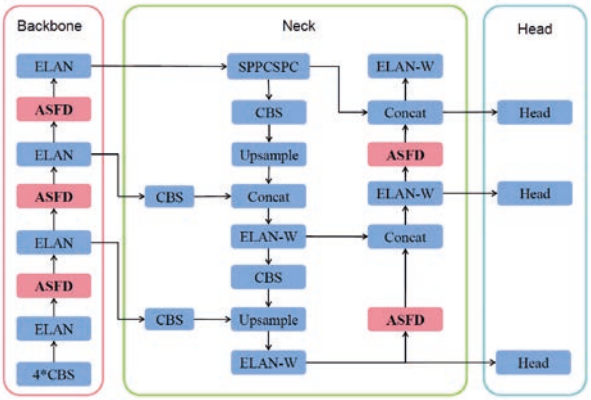


Figure 3. YOLOv7-ASFD structure. The red nodes represent our method. Other nodes are named according to the original authors of the baseline network [21]. The ASFD is usually followed by a point-wise convolution to vary the number of channels, which is not shown in the figure.

Our comparison detectors include YOLOv5 [24], YOLOv6 [25], YOLOX [26], YOLOv8 [27] and YOLOv7 baseline [22]. For each detector, we utilized the code, default hyperparameters and training strategies provided by its respective authors. Moreover, we also added the SPD-Conv [17] to YOLOv7 to compare it with our method. YOLOv7-ASFD and YOLOv7-SPD shared the same parameter settings as YOLOv7. To ensure fairness, we intentionally avoided using pre-trained weights for any of the models. This approach prevents the introduction of previously learned features during transfer learning.

Our training process involved 300 epochs for each detector, conducted on a single Tesla T4 GPU. Subsequently, we performed inference on the test set, with the input image size set to 640×640. The resulting metrics for each detector are summarized in Table 2.

In the table, $AP@0.5$ and $AP@0.75$ represent the average precision for each category at evaluated IOU thresholds of 0.5 and 0.75 respectively. Furthermore, $AP@0.5:0.95$ denotes the mean of the average precision calculated at 0.05 intervals across IOU thresholds ranging from 0.5 to 0.95. The terms AP_S, AP_M, and AP_L correspond to the average precision for small, medium, and large objects, as specified by the COCO metrics.

From the experiments, we can analyze that ASFD can bring parameter reduction and performance improvement to the model. In the YOLOv7-ASFD-tiny control group, both YOLOv6-n and YOLOX-tiny have similar or more parameters and computations than YOLOv7-ASFD-tiny, but our metrics are significantly higher than those of the former in each of them, and the $AP@0.5$ of YOLOv7-ASFD-tiny is even higher than that of YOLOX-tiny by 6.4%. YOLOv7-ASFD-tiny has a performance advantage over YOLOv7-tiny with reduced parameters. In the YOLOv7-ASFD control group, YOLOv7-ASFD compared to YOLOv7 improved both $AP@0.5$ and $AP@0.5:0.95$ by one to two percentage points while the amount of parameters was reduced by 7%, and APs of YOLOv7-ASFD was even 12.4% higher than that of YOLOv7, verifying that our method reduces the loss of information in downsampling. Compared with YOLOv6-m, which does not have much difference in the number of parameters, YOLOv7-ASFD also shows a big advantage, no matter in the overall precision, or in the precision of different sizes of objects, our index is always better than that of YOLOv6-m. For example, in $AP@0.5$,

Table 2. Comparison of YOLOv7-ASFD and other object detectors

Model	**Param**	**GFLOPS**	**AP@0.5**	**AP@0.75**	**AP@0.5:0.95**	AP_S	AP_M	AP_L
YOLOv5-n	1.9M	4.6	0.808	0.407	0.433	0.165	0.402	0.390
YOLOv6-n	4.7M	11.4	0.805	0.437	0.447	0.120	0.418	0.428
YOLOX-nano	2.2M	6.9	0.764	0.403	0.410	0.076	0.383	0.380
YOLOv8-n	3.0M	8.2	0.780	0.434	0.436	0.110	0.401	0.400
YOLOX-tiny	5.0M	15.3	0.786	0.425	0.434	0.131	0.400	0.420
YOLOv7-tiny	6.2M	13.9	0.845	0.422	0.455	0.173	0.420	0.490
YOLOv7-SPD-tiny	5.1M	11.9	0.812	0.405	0.434	0.150	0.396	0.409
YOLOv7-ASFD-tiny	4.8M	11.5	0.851	0.478	0.478	0.242	0.453	0.463
YOLOv5-s	7.2M	16.6	0.829	0.461	0.461	0.214	0.424	0.402
YOLOv6-s	18.5M	45.2	0.807	0.443	0.454	0.284	0.429	0.409
YOLOX-s	8.9M	26.8	0.812	0.471	0.467	0.226	0.445	0.425
YOLOv8-s	11.1M	28.7	0.837	0.505	0.483	0.238	0.461	0.470
YOLOv5-m	21.2M	49.2	0.834	0.493	0.472	0.269	0.442	0.432
YOLOv6-m	34.9M	85.8	0.830	0.523	0.489	0.323	0.470	0.480
YOLOX-m	25.3M	73.8	0.854	0.508	0.488	0.229	0.457	0.452
YOLOv8-m	25.7M	79.1	0.834	0.526	0.495	0.251	0.473	0.459
YOLOv7	37.3M	105.3	0.883	0.504	0.507	0.229	0.469	0.468
YOLOv7-SPD	38.0M	102.7	0.871	0.496	0.489	0.243	0.457	0.469
YOLOv7-ASFD	34.4M	102.8	0.903	0.547	0.524	0.353	0.493	0.541
YOLOv5-l	46.6M	109.6	0.835	0.524	0.487	0.312	0.464	0.462
YOLOv6-l	59.6M	150.7	0.845	0.550	0.507	0.307	0.492	0.426
YOLOX-l	54.2M	155.7	0.849	0.555	0.502	0.319	0.480	0.486
YOLOv8-l	61.0M	250.7	0.836	0.537	0.503	0.181	0.488	0.463
YOLOv5-x	86.8M	206.3	0.828	0.524	0.494	0.330	0.472	0.458
YOLOv8-x	68.2M	258.2	0.846	0.541	0.509	0.330	0.473	0.462
YOLOv7-X	71.4M	190.6	0.887	0.529	0.509	0.315	0.481	0.494
YOLOv7-SPD-X	72.6M	186.6	0.876	0.488	0.500	0.311	0.470	0.466
YOLOv7-ASFD-X	66.2M	185.2	0.898	0.606	0.550	0.331	0.527	0.577

the YOLOv7-ASFD is 90.3%, while YOLOv6-m is only 83.0%. However, the YOLOv7 baseline has advantages over YOLOv6-m in only two metrics. In the control group of YOLOv7-ASFD-X, our model is still the best overall performer in parameters and precision. Although the *APs* of our method is not very different from theirs, YOLOv6-l and YOLOv8-x have 28% higher computation than our method, which is still weaker in overall performance. And the *APs* of YOLOv7-ASFD-x is still higher than the YOLOv7 baseline. The experiments prove that ASFD is effective and feasible, and can bring performance and precision improvements to the model.

4.1.1 Time Analysis

The spatial separation stage of ASFD begins by calculating the offsets and slices, with a time complexity of $O(1)$. Subsequently, the time complexity of adding elements based on position is $O(C_{in} \times H \times W)$. H and W are represent the dimensions of the feature map, while C_{in} denotes the number of channels. Thus, the overall time complexity for the first stage is $O(C_{in} \times H \times W)$. In the second stage, the time complexity for the five convolutional layers in the spatial perception and feature activation branches is $O(C_{in}^2 \times C_{mid}^2 \times H \times W \times K^2)$. C_{mid}^2 represents the middle channels, and K^2 denotes the size the convolution kernel. Operations such as global average pooling, ReLU, sigmoid, and softmax have an average time complexity of $O(C_{mid}^2 \times H \times W)$. Multiplying the attention map with the intermediate feature map also has a time complexity of $O(C_{in} \times H \times W)$. Additionally, a 1×1 convolution is typically used in practical scenarios to adjust the number of channels in the final downsampling result. Therefore, the overall time complexity of the second stage is $O(C_{in}^2 \times C_{out}^2 \times H \times W \times K^2)$. While this is comparable to the time complexity of strided convolution, the use of multiple convolutional layers means the actual computation time is not as efficient as a single strided convolution. However, since our algorithm often employs convolutions with constant output channels or feature map sizes of 1×1, the actual runtime does not increase significantly compared to strided convolution. As shown in Table 4.1.1, our proposed method did not significantly improve the time. However, as shown in Table 2, our method achieves a 10% to 20% reduction in the number of parameters compared to the baseline. Notably, the smaller the scale of the model, the greater the percentage of parameter reduction achieved by our approach.

Table 3. The inference speed of each image in different scale models (ms).

Model	Tiny	Standard	X
YOLOv7	13.6	43.0	80.5
YOLOv7+ASFD	14.0 ↑ 2.9%	49.1 ↑ 14.2%	87.9 ↑ 9.2%

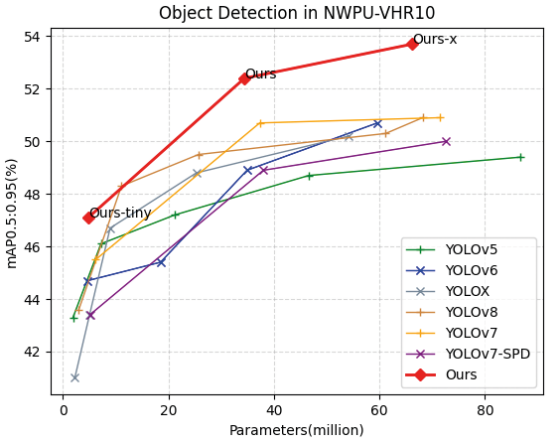


Figure 4. Parameter–precision curve. The x-axis is a parameter, measured in millions. And the y-axis represents the average precision of the IOU threshold within the range of 0.5 to 0.95, measured in percentage.

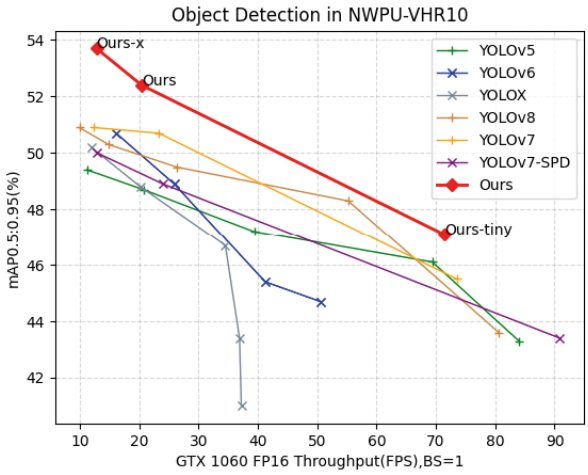


Figure 5. Throughput–precision curve. The x-axis represents the throughput of NVIDIA 1060 GPU with batchsize of 1, measured in FPS. And the yaxis represents the average precision, measured in percentage.

To more accurately compare the performance of different methods under identical inference speed and parameter count, we plotted the parameter-precision change curve and the throughput-precision change curve according to the change of precision with the parameters of the model and the change of precision with the detection speed, respectively. Figs. 4 and 5 demonstrate that YOLOv7-ASFD, which integrates our downsampling module, achieves higher precision performance with the same parameters and throughput compared with YOLOv7 baseline. At the same inference speed, we achieved higher accuracy, indicating a deliberate trade-off of some time to gain improved performance. We consider it is worthwhile.

4.1.2 Visualization of Results

We randomly selected some images from the test set to visualize the better detection results of the network with the addition of ASFD module. YOLOv7-ASFD integrates the ASFD module on the YOLOv7 baseline, so a comparative analysis between the two confirms the efficacy of the ASFD. In addition, YOLOv7 performed well in the above experiment, so we compared the detection results of YOLOv7-ASFD and YOLOv7.

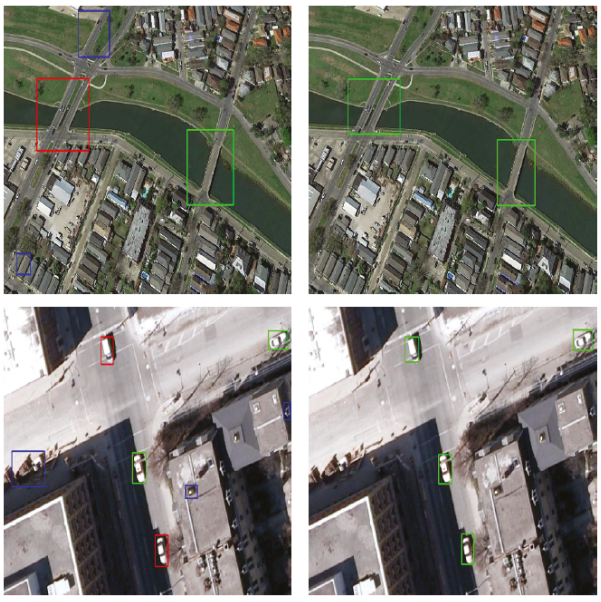
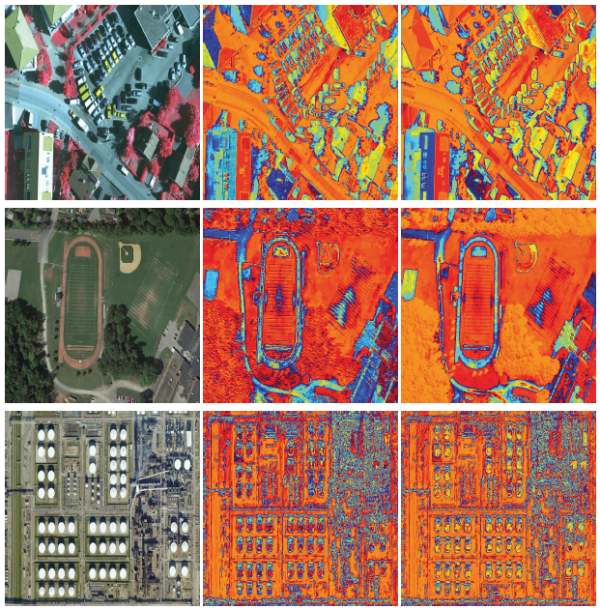


Figure 6. Visualization of object detection prediction results. The left image is the detection results of YOLOv7, and the right image is the detection results of YOLOv7-ASFD.

The green boxes in Fig. 6 indicate correctly predicted objects, while the blue boxes represent incorrect predictions, and the red boxes highlight missed objects. In Fig. 6(a), YOLOv7 fails to detect the bridge and mistakenly classifies the road as a bridge. It also misses several vehicles and incorrectly detects roadside debris as vehicles. However, in Fig. 6(b), YOLOv7-ASFD avoids these errors, and the predictions align more closely with the ground truth. Analyzing the detection results, we observe that YOLOv7 performs worse than YOLOv7-ASFD, particularly for small objects, due to information loss during downsampling. Additionally, the road and bridge are visually similar, making it challenging for the detector to distinguish between them. YOLOv7-ASFD, however, correctly identifies the bridge by leveraging inter-region relationships (e.g., distinguishing water from land). In conclusion, our method preserves more feature map information during downsampling while emphasizing the relationships between regions, leading to improved detection performance.



(a) Origin (b) YOLOv7 (c) YOLOv7-ASFD

Figure 7. Visualization of heatmaps of object detection.

We also visualize the heatmap during the forward propagation of the model. As shown in Fig. 4.1.2, YOLOv7-ASFD exhibits more focused and stronger attention on objects. For example, YOLOv7's attention on vehicles and storage

tanks is weak and incomplete, whereas YOLOv7-ASFD, with region-based attention relationships, is able to construct comprehensive perceptions by capturing contextual relations both within and between regions. For large objects such as playgrounds, YOLOv7 struggles with long-distance dependencies, resulting in weak attention. In contrast, YOLOv7-ASFD strengthens attention across the entire playground by transferring long-distance information through constructed inter-region relationships. ASFD enhances the construction of short-distance dependencies through intra-region attentional relationships and improves long-distance dependencies via inter-region attentional relations, thus boosting the model's ability to capture both small and large objects effectively.

4.2 ASFD Performance in Classification

The experiments above validate that ASFD achieves better performance with fewer parameters during downsampling. Next, we assessed the generalizability and robustness of our method.

We evaluated the performance of ASFD and downsampling methods commonly used by some current networks on the Tiny ImageNet dataset [28]. This dataset is a subset of the ILSVRC-2012 classification dataset from Stanford University's course project on cs231N. It comprises 200 classes, each with 500 training images, 50 validation images, and 50 test images, all at a resolution of 64×64. Our evaluation includes four representative networks: VGG16 [1], ResNet18 [2], MobileNet [29], and FasterNet [30]. In addition, we used common downsampling methods such as strided convolution, MaxPool, AvgPool, and SoftPool [9] which is a improved pooling method, as well as our comparison methods Spd-Conv [17] and FouriDown [16]. We implemented these networks and methods using the code provided by the respective authors.

First, we integrated ASFD and other downsampling methods into four baseline models. The same baseline models were then grouped for comparison based on the Top-1 accuracy of each group's classification predictions. Consistent parameter settings, data augmentation techniques, learning rates, and optimization strategies were applied across all comparisons within each group, with the goal of investigating performance differences among various downsampling methods. The training and evaluation were conducted in an NVIDIA A100 GPU environment, and the experimental results are presented in Fig. 8. The baseline results are shown in the first column of each figure.

After applying the ASFD downsampling method to VGG16, ResNet18, MobileNet, and FasterNet, their classification accuracies significantly improved.

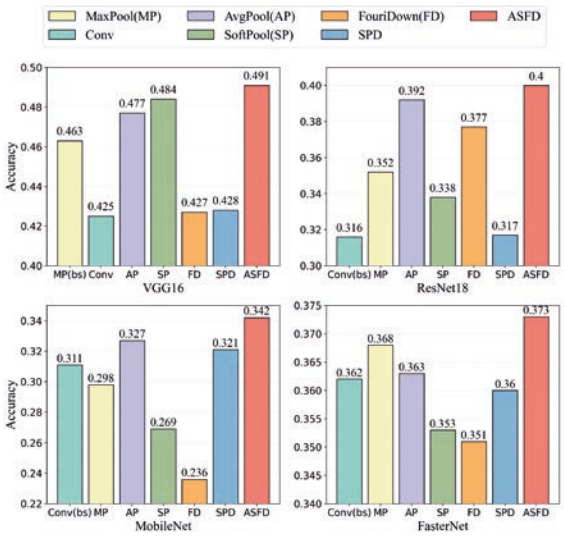


Figure 8. Comparison of Top-1 accuracy of different downsampling methods.

In the VGG16 control group, our method achieved higher accuracy than all other downsampling methods, which surpasses SoftPool by 0.7%. In the ResNet18 control group, our method achieved a Top1 accuracy of 40.0%, which is 0.8% higher than the second-best AvgPool method. Although other methods also showed improvements compared with the baseline, our model demonstrated the most substantial enhancement. The reason for the higher boost is that the ResNet18 baseline uses convolution to downsample. Owing to the small pixel size of the Tiny ImageNet dataset images and the high complexity of the ResNet18 model, multiple convolutions introduce redundant feature representations. Consequently, the model tends to overfit and struggles to learn deeper feature representations. By contrast, ASFD reduces the number of parameters in the model, which enables it to avoid overfitting. In the MobileNet control group, our approach surpasses the second-best AvgPool method by a margin of 1.5%. MobileNet extracts less feature information than normal convolution due to its depthwise separable convolu-

tion architecture. Consequently, if pooling is used for downsampling, then more effective information may be lost. However, ASFD effectively alleviates this problem. Our method retains feature map information in a fused manner, which significantly contributes to improving the classification performance of lightweight networks. We have a 1.1% improvement in classification accuracy over the FasterNet baseline. In this group, insignificant differences in classification accuracy are observed across various methods. The reason is that FasterNet effectively extracts information through partial channel convolution, which suppresses redundant information. The reasonable network structure also makes the feature representation more complete, which results in minimal differences in the effect of each downsampling method. The abovementioned experiments show that our proposed ASFD method serves as a plug-and-play downsampling module, is universally applicable to different models, andoutperforms previous downsampling methods.

4.3 Ablation Experiment

The ASFD module consists of two stages, one is spatial separation and the other is adaptive spatial fusion. Among them, adaptive spatial fusion contains two branches, a spatial fusion branch (Spatial Perception) and a channel activate branch (Feature Activation). To evaluate the impact of these branches on performance improvement, we trained models using spatial separation, spatial perception, feature activation, and all simultaneously. Our experiments, based on object detection, demonstrates that these branches contribute to downsampling improvement. Detailed results are provided in Table 4.3.

Firstly, spatial separation contributes to an improvement in overall precision. However, as shown in the table, its effectiveness diminishes in scenarios where the IOU threshold is high. Its performance is particularly limited when detecting smaller objects, as evidenced by the decreased performance observed for $AP@0.75$ and AP_S. The spatial separation stage focuses only on isolating local features in the feature map. In deep networks, such isolation can disrupt the spatial structure and semantic integrity of these features. Therefore, under conditions of high IOU requirements and suboptimal small object detection performance, additional measures are necessary. At this stage, the incorporation of a second process becomes essential. This stage adaptively learns the strengths and weaknesses of intra-regional feature relationships, as well as inter-regional feature correlations.

On the basis of the spatial separation stage, spatial perception branche and feature activation branch independently contribute to an increase in the detection precision of the network. When integrating spatial perception into network, $AP@0.75$ substantially improves. It reaches 52.7%, which is a 2.3% improvement compared with the baseline. Similarly, integrating Feature Activation also has a positive effect. Secondly, both branches significantly enhance the performance of the first stage, particularly in AP_S, where precision improves by an impressive 6% to 8%. The simultaneous use of both branches enhances the network's adaptive learning capability. All metrics exhibit significant improvement compared to the use of the branches individually.This observation suggests that the two branches are complementary and mutually reinforcing, without adversely affecting each other's effectiveness. Owing to the existence of these two branches, our model can more accurately discern the contour and texture details of objects. This capability aligns precisely with the feature expression that the detector needs to learn under high IOU thresholds, which allows our method to cater to finer detection requirements. When comparing medium and large objects, ASFD exhibits the most significant accuracy improvement for small objects. We attribute this to the ability of ASFD to largely retain all the information from feature maps in downsampling. It selectively preserves feature information that contributes most to the region in an adaptive manner. However, even relatively weak information is retained in the result after downsampling through fusion. As a result, the information loss is significantly reduced compared with conventional convolution.

We also replaced the spatial separation stage with strided convolutions for downsampling, followed by an adaptive spatial fusion stage. Experimental results indicate that while the overall precision of the network shows a relative improvement, the increase is not substantial, and the precision for small objects decreases compared to the baseline. We attribute this to the absence of the initial stage, which is crucial for preserving complete informa-

Table 4. Effect of two branches of ASFD in NWPU-VHR10.

Method	AP@0.5	AP@0.75	AP@0.5:0.95	APS	APM	APL
YOLOv7	0.883	0.504	0.507	0.23	0.47	0.464
YOLOv7+SS (Spatial Separation)	0.885 (+0.2%)	0.496 (-0.8%)	0.508 (+0.1%)	0.186 (-4.4%)	0.465 (-0.5%)	0.494 (+3.0%)
YOLOv7+SS+SP (Spatial Perception)	0.885 (+0.2%)	0.527 (+2.3%)	0.509 (+0.2%)	0.293 (+6.3%)	0.484 (+1.4%)	0.501 (+3.7%)
YOLOv7+SS+FA (Feature Activation)	0.890 (+0.7%)	0.533 (+2.9%)	0.511 (+0.4%)	0.319 (+8.9%)	0.490 (+2.0%)	0.517 (+5.3%)
YOLOv7+SP+FA	0.898 (+1.5%)	0.512 (+0.8%)	0.511 (+0.4%)	0.217 (-1.3%)	0.488 (+1.8%)	0.499 (+3.5%)
YOLOv7+SS+SP+FA	0.901 (+1.8%)	0.545 (+4.1%)	0.523 (+1.6%)	0.353 (+12.3%)	0.491 (+2.1%)	0.536 (+7.2%)

tion. The inherent loss in convolution is challenging to mitigate, thereby limiting significant performance gains. The above experiments demonstrate that our method functions as an internally interdependent whole.

5 Conclusion

We propose a novel approach, Adaptive Separation Fusion Downsampling (ASFD), to address the challenge of information loss commonly caused by existing downsampling methods. ASFD fuses feature representations within submap regions, enabling different regions to focus on their respective attention areas. This approach mirrors the way humans analyze images, enhancing the model's ability to capture meaningful features. Notably, ASFD is a unified method that can be seamlessly applied across various models. Extensive evaluations show that networks employing ASFD achieve superior classification and detection precision in both image classification and object detection tasks.

References

[1] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778, 2016.

[3] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 580–587, 2014.

[4] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18, pp. 234–241, Springer, 2015.

[5] W. Zhang, Z. Hong, L. Xiong, Z. Zeng, Z. Cai, and K. Tan, "Sinextnet: A new small object detection model for aerial images based on pp-yoloe," Journal of Artificial Intelligence and Soft Computing Research, vol. 14, no. 3, pp. 251–265.

[6] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in European conference on computer vision, pp. 213–229, Springer, 2020.

[7] Y.-L. Boureau, F. Bach, Y. LeCun, and J. Ponce, "Learning mid-level features for recognition," in 2010 IEEE computer society conference on computer vision and pattern recognition, pp. 2559–2566, IEEE, 2010.

[8] Y.-L. Boureau, J. Ponce, and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition," in Proceedings of the 27th international conference on machine learning (ICML-10), pp. 111–118, 2010.

[9] A. Stergiou, R. Poppe, and G. Kalliatakis, "Refining activation downsampling with softpool," in Proceedings of the IEEE/CVF international conference on computer vision, pp. 10357–10366, 2021.

[10] M. D. Zeiler and R. Fergus, "Stochastic pooling for regularization of deep convolutional neural networks," 2013.

[11] D. Yu, H. Wang, P. Chen, and Z. Wei, "Mixed pooling for convolutional neural networks," in Rough Sets and Knowledge Technology: 9th International Conference, RSKT 2014, Shanghai, China, October 24-26, 2014, Proceedings 9, pp. 364–375, Springer, 2014.

[12] C. Gulcehre, K. Cho, R. Pascanu, and Y. Bengio, "Learned-norm pooling for deep feedforward and recurrent neural networks," in Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2014, Nancy, France, September 15-19, 2014. Proceedings, Part I 14, pp. 530–546, Springer, 2014.

[13] S. Zhai, H. Wu, A. Kumar, Y. Cheng, Y. Lu, Z. Zhang, and R. Feris, "S3pool: Pooling with stochastic spatial sampling," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4970–4978, 2017.

[14] Z. Gao, L. Wang, and G. Wu, "Lip: Local importance-based pooling," in Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 3355–3364, 2019.

[15] J. Zhao and C. G. M. Snoek, "Liftpool: Bidirectional convnet pooling," 2021.

[16] Q. Zhu, J. Huang, N. Zheng, H. Gao, C. Li, Y. Xu, F. Zhao, et al., "Fouridown: factoring down-sampling into shuffling and superposing," Advances in Neural Information Processing Systems, vol. 36, 2024.

[17] R. Sunkara and T. Luo, "No more strided convolutions or pooling: A new cnn building block for low-resolution images and small objects," in Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 443–459, Springer, 2022.

[18] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: analysis, applications, and prospects," IEEE transactions on neural networks and learning systems, vol. 33, no. 12, pp. 6999–7019, 2021.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," Communications of the ACM, vol. 60, no. 6, pp. 84–90, 2017.

[20] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2020.

[21] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "Cbam: Convolutional block attention module," in Proceedings of the European conference on computer vision (ECCV), pp. 3–19, 2018.

[22] C.-Y. Wang, A. Bochkovskiy, and 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, pp. 7464–7475, 2023.

[23] H. Su, S. Wei, S. Liu, J. Liang, C. Wang, J. Shi, and X. Zhang, "Hq-isnet: High-quality instance segmentation for remote sensing imagery," Remote Sensing, vol. 12, no. 6, p. 989, 2020.

[24] G. Jocher, "YOLOv5 by Ultralytics," May 2020.

[25] C. Li, L. Li, H. Jiang, K. Weng, Y. Geng, L. Li, Z. Ke, Q. Li, M. Cheng, W. Nie, Y. Li, B. Zhang, Y. Liang, L. Zhou, X. Xu, X. Chu, X. Wei, and X. Wei, "Yolov6: A single-stage object detection framework for industrial applications," 2022.

[26] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "Yolox: Exceeding yolo series in 2021," 2021.

[27] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLO," Jan. 2023.

[28] Y. Le and X. Yang, "Tiny imagenet visual recognition challenge," CS 231N, vol. 7, no. 7, p. 3, 2015.

[29] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017.

[30] J. Chen, S.-h. Kao, H. He, W. Zhuo, S. Wen, C.-H. Lee, and S.-H. G. Chan, "Run, don't walk: Chasing higher flops for faster neural networks," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12021–12031, 2023.

Xia Ji Ph.D., associate professor. Her research interests include artificial intelligence, machine learning, and intelligent information processing. She has taken charge of numerous school-enterprise cooperation projects, one key research and development program of Anhui Province, two provincial natural science foundation projects of Anhui higher education institutions, and participated in three major projects of the National Natural Science Foundation of China as well as a number of provincial natural science foundation projects.
https://orcid.org/0000−0002−2820−0405

Jinglong Chang, He is a third-year master's student at Anhui University's School of Computer Science and Technology. He is a member of Anhui University International Brain Science Engineering Center. His main research interests are computer vision and object detection.
https://orcid.org/0009-0005-5363-4271

Yapeng Ji, He is a third-year master's student at Anhui University's School of Computer Science and Technology and member of Anhui University International Brain Science Engineering Center. His main research interests is small object detection.
https://orc id.org/0009-0001-7287-1302