DSIC: DYNAMIC SAMPLE-INDIVIDUALIZED CONNECTOR FOR MULTI-SCALE OBJECT DETECTION

Zekun Li[1,2*], *Yufan Liu*[1,2*], *Bing Li*[1,3†], *Weiming Hu*[1,2,4], *Yanan Miao*[5], *Hong Zhang*[5]

[1]National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences, [3]PeopleAI Inc
[4]CAS Center for Excellence in Brain Science and Intelligence Technology, [5]CNCERT/CC.
(lizekun2018,yufan.liu)@ia.ac.cn,(bli,wmhu)@nlpr.ia.ac.cn,miaoyn@cert.org.cn,zhangh@isc.org.cn

ABSTRACT

Although object detection has reached a milestone recently, the scale variation is still the key challenge. Integrating multi-level features is presented to alleviate the problems, like Feature Pyramid Network (FPN) and its improvements. However, the specifically designed architectures and fixed data flow paths of these methods are not flexible for feature fusion, especially when fed with various samples. To overcome the limitations, we propose a Dynamic Sample-Individualized Connector (DSIC) for multi-scale object detection, which dynamically adjusts network connections to fit different samples. In particular, DSIC consists of two components: Intra-scale Selection Gate (ISG) and Cross-scale Selection Gate (CSG). With the help of the presented gate operator, ISG adaptively extracts proper multi-level features from backbone as the inputs of feature integration. CSG automatically activates informative data flow paths based on the extracted multi-level features. These two components are both plug-and-play and can be embedded in any backbone. Experimental results demonstrate that the proposed method outperforms the state-of-the-arts.

***Index Terms*—** Object Detection, Scale Variation, Gate, Sample-Individualized Connector

1. INTRODUCTION

Object detection has been explored for many years as a foundation in computer vision. With the great development of deep learning, object detection has achieved remarkable progress. Plenty of excellent detectors [1, 2, 3, 4, 5, 6] are proposed to improve the performance and show extraordinary results on public benchmarks such as MS-COCO [7].

However, there are still some problems limiting the performance of detectors. Scale variation is one of the most challenging problems. Feature Pyramid Network (FPN) [8] is an effective method to alleviate this problem by merging features at adjacent levels to construct a top-down pyramid

*Equal contribution. † Corresponding author.

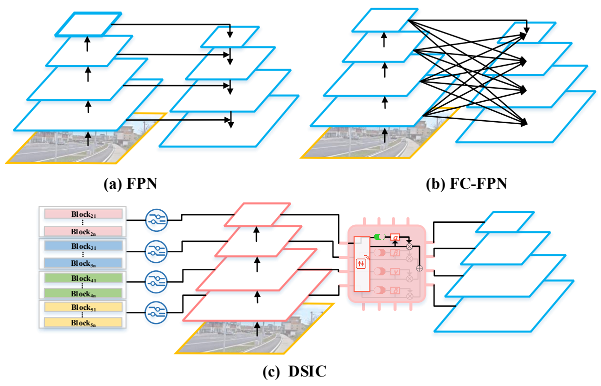

Fig. 1. Illustrations of (a) the classical FPN, (b) Fully-Connected FPN (FC-FPN) and (c) our method Dynamic Sample-Individualized Connector (DSIC). FPN and FC-FPN represent the different cases of connection which use the same output of backbone as input, constantly. Our DSIC can learn to select input from backbone and activate different data flow paths as connection according to the input sample.

(seen as Fig 1 (a)). More recently, some studies [9, 10, 11] have been proposed to improve the connection design in FPN. Nevertheless, the manual designed architectures have fixed connections, ignoring the diversity brought by different samples. And the interaction across multi-level features is not adequate, because the specific connections cannot be the optimal case in various situations. Fully-Connected Feature Pyramid Network (FC-FPN), depicted in Fig 1 (b), can be seen as the universal set that includes all connection methods between the bottom-up and top-down pyramid. It uses full connection to enhance the feature representation without manual design. But meanwhile, redundancy and noise are brought in, which indicates that it is unnecessary to activate all connections. Furthermore, both series of FPN and FC-FPN are fixed, which are not friendly to different samples.

In this paper, in order to alleviate aforementioned problems, we propose a novel and effective module called Dynamic Sample-Individualized Connector (DSIC). Different from the methods mentioned above (i.e., Fig 1(a) and (b)), DSIC dynamically selects the proper input and connections, like Fig 1(c) illustrated. It avoids fixed specific design, and automatically adjusts the connection and feature interaction process, to fit different samples. In particular, we firstly present a gate operator as the basic element of DSIC, which controls the state of data flow path. Given control signals, it can connect or disconnect a data flow path, even further enhance or suppress the data flow. Based on the gate operator, DSIC is constructed, consisting of two components: Intra-scale Selection Gate (ISG) and Cross-scale Selection Gate (CSG). ISG aims to explore what is the proper input of feature integration which adaptively extracts multi-level features with sufficient information from backbone. CSG devotes to learning what is the proper connections among multi-level features when fed with different samples. It automatically activates informative data flow paths based on the multi-level features. These two components are both plug-and-play and can be embedded in any backbones separately or jointly. DSIC shows superiority when compared with the state-of-the-arts and other feature integration methods.

In summary, this work makes the following main contributions:

- We propose a novel plug-and-play block called DSIC to alleviate the scale-variant problem. To control the connection of data flow path dynamically, we firstly present the gate operator as the basic element of DSIC, which is the core operation of the proposed method.
- We propose Intra-scale Selection Gate (ISG) and Cross-scale Selection Gate (CSG) to constitute DSIC, taking advantage of the presented gate operator. ISG aims to adaptively extracts multi-level features from backbone as input, while CSG devotes to automatically activating data flow paths. Each of these components can be separately or jointly embedded into any backbones.
- We evaluate the proposed framework on MS-COCO 2017 and it shows the superiority when compared with state-of-the-arts. The ablation experiments validate the effectiveness of each module of DSIC.

2. RELATED WORKS

2.1. Multi-level feature extraction

Scale variation of object instances is a gargantuan obstacle in object detection. The integration of multi-level features is beneficial to mitigate such problem. FPN [8] was designed to fuse features through a top-down pyramid. After that, a series of works were presented with improved structures based on FPN, to further enhance the performance. For example, PANet [9] improved FPN by adding a new bottom-up structure after the feature pyramid to shorten information path. FPG [12] was proposed utilizing a deep multi-pathway feature pyramid that repeated the fusion process in different directions. Although the methods above have obtain some progress, there are still some problems. The multi-level features extracted from backbone and data flow paths of integration in these networks are fixed, when different inputs and features are being processed. This is not flexible.

2.2. Dynamic Mechanisms

More recently, dynamic mechanisms have been explored to improve the performance of models in computer vision tasks, which adaptively adjust some variables or settings of the network. Some methods use dynamic mechanism to adjust the network configurations. DRConv [13] assigns filters to learning appointed spatial areas that achieved better performance through obtaining rich and diverse spatial information. Other methods dynamically learn to set the hyper-parameters. Dynamic R-CNN [14] was proposed to alleviate the inconsistency between the hyper-parameters and training procedure, by automatically adjusting the IoU threshold and the parameters of loss function. Nevertheless, the existing dynamic mechanisms only focus on training or network configurations adjustment, ignoring the feature settings and data flow path selection. We propose a new dynamic sample-individualized connector that can select the superior features in multi blocks from backbone and the better multi-level feature integration strategy when processing different samples dynamically.

3. THE PROPOSED METHOD

3.1. Foundation

Pipeline Existing feature pyramid module is shown in Fig 2 (a), which aims to integrating multi-level features to solve multi-scale object detection. In this module, $\{C_2, C_3, C_4, C_5\}$ represent the inputs of the feature pyramid. Note that, C_i is extracted from the output of the last block in the i-th stage of the backbone. In addition, $\{P_2, P_3, P_4, P_5\}$ represent the new feature maps after feature integration. P_k denotes the new feature map at the k-th level. Due to the success of alleviating the scale variation problem, feature pyramid module is widely used in the recent years. However, this module has fixed the input and connection data flow path.

We utilize DSIC to dynamically construct the feature integration module, as illustrated in Fig 2(b). In particular, the proposed DSIC comprises two components: ISG and CSG. ISG is put in the left of the feature integration module, utilized to select the proper features in each stage of the backbone. Note that $Block_{ij}$ denotes the j-th block in the i-th stage. Besides, CSG is leverage to replace the original connection, to obtain a proper connection cross variant-scale features.
Gate Operator The gate operator is the basic element of DSIC, which controls the data flow path. In detail, given data flow $\mathbf{X} \in \mathbb{R}^{c\times h\times w}$ as input, the gate operator $\mathcal{G}(\cdot)$ can be formulated as:

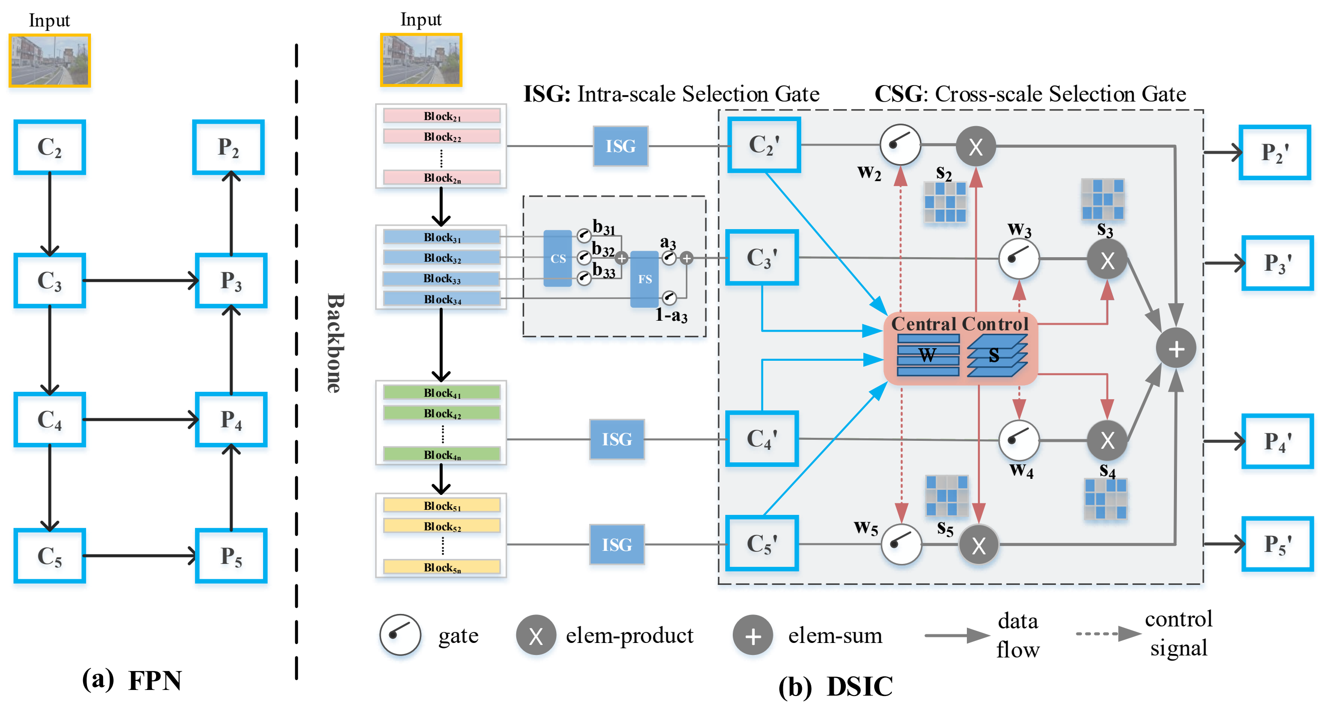

Fig. 2. (a) The conventional feature pyramid network. (b) The framework of the proposed DSIC.

$$\mathcal{G}(\varepsilon, \mathbf{X}) = \sigma[\Psi(\varepsilon, \sum_{\Phi\in\mathcal{F}} \Phi(\mathbf{X}))], \qquad (1)$$

in which $\varepsilon \in \mathbb{R}^{m\times1\times1}$ denotes the gate control signal. It assists the gate to determine whether to open or close, and further enhance or suppress the data flow when the gate is open. $\mathcal{F}$ represents the set of functions, including a series of convolutions, making the input compatible with the gate. $\Psi(\cdot)$ denotes the Hadamard product and $\sigma(\cdot)$ represents the mode selection of gate operator by activation function. In this manner, given different data and control signals, the gate shows different states so that the inputs and connection data flow paths are dynamically changed.

3.2. Intra-scale Selection Gate (ISG)

The input with sufficient information is crucial to feature integration. The conventional FPN only extracts the feature maps from each stage's last block as input. This design is rigid, in which only single block's output is used while the information of the rest is lost. Contrastively, we propose ISG to dynamically select the intra-scale information in backbone from coarse to fine and obtain an input with adequate information.

As shown in Fig 2 (b), ISG is comprised by Coarse Selection (CS) and Fine Selection (FS). Because of the stronger representation of last block, CS firstly selects the useful information from $(n-1)$ former blocks, to compensate the information that the last block lacks. FS further selects a proper fusion of the last block's output and the complementary data.

In particular, the $(n-1)$ former blocks $\{B_{ij}\}_{j=1}^{j=n-1}$ in the i-th stage are fed into CS, and then the corresponding control signals b_{ij} are obtained to adjust the states of the gates:

$$\{b_{i1}, ...b_{ij}\} = \sum_{\Phi\in\mathcal{I}} \Phi\,\{B_{i1}, ...B_{ij}\}, \quad j = (1, ...n-1), \quad (2)$$

where $\mathcal{I}$ denotes set of integration operators, including channel-wise concatenation, a series of convolutions and poolings. Note that the parameters of CS in each stage are non-shared. After that, given outputs of blocks and control signals, a series of gates control the data flow paths and obtain:

$$B_i^{CS} = \sum_{j=1}^{n-1} \mathcal{G}(b_{ij}, B_{ij}). \qquad (3)$$

The gate operators $\mathcal{G}(\cdot)$ control whether the data flows from some blocks are needed currently. B_i^{CS} denotes the result of selection from previous output. After the process of CS, the results are fed into FS to select an integration of $B_{i,n}$ and B_i^{CS}. Similarly, considering $B_{i,n}$ and B_i^{CS}, FS computes the gate control signals a_i:

$$a_i = Max[tanh(F_{gap}(B_i^{CS}) + F_{gmp}(B_i^{CS})), 0], \qquad (4)$$

where $tanh$ denotes the Tanh activation operator and F_{gap} and F_{gmp} means global average-pooling and global max-pooling, respectively. Here, $a_i \in \mathbb{R}^{c\times1\times1}$ is the control signals of B_i^{CS} that determines which channels should pass in the gate operator. After the computation of FS, the dynamically selected input of feature integration C_i' can be obtained:

$$C_i' = \mathcal{G}(a_i, B_i^{CS}) + \mathcal{G}((1-a_i), B_{i,n}). \qquad (5)$$

Instead of the fixed input of FPN, we rethink the effect of all blocks in backbone and rearrange the retention and abandon of information. The proposed ISG achieves coarse-to-fine selection, which can dynamically extract the useful information as input according to various samples in backbone.

3.3. Cross-scale Selection Gate (CSG)

In multi-scale object detectors, feature integration module merges multi-level feature maps $\{C'_2, C'_3, C'_4, C'_5\}$ via a lateral connections, in order to obtain a feature pyramid $\{P'_2, P'_3, P'_4, P'_5\}$ with rich semantics at all levels. As shown in Fig 1(b), the full connection in FC-FPN can be seen as the universal set of the lateral connections. This case seems to fully integrate features, but there exists large redundancy. On the other hand, the conventional FPN (seen in Fig 2(a)) is the subset of FC-FPN. It only connects the features at the same level from bottom-up pyramid to top-down pyramid in lateral direction. Both of them are fixed, when fed with different samples.

Different from the cases above, we propose CSG to dynamically activate useful data flow paths, and obtain a flexible connection case when meeting various samples. In detail, as illustrated as Fig 2(b), taking $\{C'_2, C'_3, C'_4, C'_5\}$ as input, the central control unit $\mathscr{CCU}(\cdot)$ firstly generates gate control signals $\mathbf{w_k} = \{w_{ik}\}_{i=2}^{5}$ and pixel-wise selection maps $\mathbf{s_k} = \{s_{ik}\}_{i=2}^{5}$:

$$\{\mathbf{w_k}, \mathbf{s_k}\} = \mathscr{CCU}(\{\mathbf{M_k}\}) \qquad k = (2, 3, 4, 5), \tag{6}$$

$$M_{ik} = \begin{cases} F_{down}^{(k-i)}(C'_i) & i < k \\ C'_i & i = k \\ F_{up}^{(i-k)}(C'_i) & i > k \end{cases}. \tag{7}$$

Note that $\mathbf{M_k} = \{M_{ik}\}_{i=2}^{5}$ are the input feature maps with unified resolution. Specifically, to compute M_{ik}, higher-resolution features are down-sampled by the 3×3 convolution and lower-resolution features are up-sampled by bilinear interpolation with appropriate scale factors. In Equation 7, F_{down}^{k-i} is 3×3 convolution operators with$(k-i)$times and F_{up}^{i-k} is bilinear interpolation with scale factor of $(i-k)$.

More detailedly, inside the central control unit, the gate control signals $\{w_{ik}\}_{i=2}^{5}$ are computed as:

$$\{w_{ik}\}_{i=2}^{5} = \sum_{\Phi \in \mathscr{O}_1} \Phi(\{M_{ik}\}_{i=2}^{5}). \tag{8}$$

$\mathscr{O}_1$ indicates the operation set, including convolutions and activation functions. The gate control signals w_k are utilized to adjust the corresponding gate units, so that the useful data flow path can be connected. After the data flow passes the gate, a pixel-wise selection map generated by the central control unit is leveraged to activate the data flow spatially, and further make features from different levels consistent. When it comes to different data flows, the generated selection maps have different activated pixels and significant regions. In particular, inside the central control unit, the pixel-wise selection maps $\mathbf{s_k} = \{s_{ik}\}_{i=2}^{5}$ at the k-th level:

$$\{s_{ik}\}_{i=2}^{5} = \sum_{\Phi \in \mathscr{O}_2} \Phi(\{M_{ik}\}_{i=2}^{5}) \tag{9}$$

$\mathscr{O}_2$ contains a modulus operator that normalize M_{ik} spatially and a series of operations, including convolutions, activation functions and element-wise sum. Subsequently, the final feature pyramid output can be calculated as:

Table 1. Comparison with baselines on $val2017$. "$\surd$" means the baseline models integrated with our connector and others mean the baseline models integrated with FPN by default.

Method	Backbone	DSIC	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
FCOS	ResNet-50		36.6	55.7	38.8	20.7	40.1	47.4
		√	**37.7**	**56.5**	**40.0**	**22.1**	**41.2**	**48.8**
	ResNet-101		39.2	**58.8**	42.1	22.9	42.8	**51.6**
		√	**40.0**	58.4	**43.1**	**23.9**	**43.9**	51.2
Faster R-CNN	ResNet-50		36.3	58.4	39.1	21.5	40.0	46.6
		√	**38.3**	**59.7**	**41.7**	**22.5**	**41.7**	**49.5**
	ResNet-101		38.3	60.0	41.8	22.8	42.6	49.5
		√	**39.6**	**61.4**	**43.2**	**22.8**	**43.1**	**50.0**
Mask R-CNN	ResNet-50		37.3	59.2	40.4	22.3	40.6	46.3
		√	**38.8**	**60.5**	**42.1**	**22.5**	**41.8**	**48.5**
	ResNet-101		39.4	60.9	43.1	22.9	43.9	51.1
		√	**40.6**	**60.7**	**44.5**	**24.6**	**44.4**	**51.7**

$$P'_k = \left[\sum_{i=2}^{5} (\mathcal{G}(w_{ik}, M_{ik}) \cdot s_{ik}) \right], \tag{10}$$

where P'_k denotes the output at the k-th level, containing rich information at all levels.

Thus, CSG takes features at all levels into consideration, and automatically selects different connections when meeting different samples. This connection case can eliminate redundant information and achieve a better performance, compared with the full connection case. Besides, the pixel-wise selection can further refine the passed data flow spatially.

4. EXPERIMENTS

4.1. Settings

Our experiments are implemented on MS-COCO 2017 which is a challenging and credible dataset containing 80 object categories. It consists of 115k images for training ($train2017$), 5k images for validation ($val2017$) and 20k test-dev images ($testdev$). The training process is performed on $train2017$, and ablation experiments and final results are evaluated on $val2017$ and $testdev$, respectively. The performance is evaluated by standard COCO-style Average Precision (AP) metrics. In order to ensure the fairness of the experiment comparisons, we implement our method and re-implement baseline methods based on PyTorch [15] and mmdetection [16].

4.2. Performance

Comparison with the baseline. As shown in Table 1, DSIC achieves consistent improvement overall all baseline detectors, which brings the definite improvements on various public backbone and different detectors. The better performance proves the generalization and robustness ability of DSIC.
Comparison with other feature integration modules. As shown in Table 2, we compare our method with other different feature integration modules with same configs. It is obvious that DSIC provides the better improvement when

Table 2. Comparison with other feature integration methods on Faster R-CNN with ResNet-50 on $val2017$.

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
FPN	36.3	58.4	39.1	21.5	40.0	46.6
FC-FPN	37.5	59.1	40.3	21.8	41.4	48.7
PANet	37.7	59.5	40.7	21.8	41.5	48.9
FPG	38.0	59.4	41.2	22.1	40.7	46.4
FPT	38.0	57.1	38.9	20.5	38.1	**55.7**
Ours	**38.3**	**59.7**	**41.7**	**22.5**	**41.7**	49.5

Table 4. Effectiveness of each component. ISG: Intra-scale Selection Gate, CSG: Cross-scale Selection Gate.

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
baseline	36.3	58.4	39.1	21.5	40.0	46.6
baseline+ISG	37.6	59.0	40.5	21.7	40.9	49.0
baseline+CSG	37.8	59.1	41.0	21.8	40.8	48.9
baseline+ISG+CSG	**38.3**	**59.7**	**41.7**	**22.5**	**41.7**	**49.5**

compared with common feature pyramid networks including PANet [9], FPT [17] and FPG [12]. DSIC can dynamically select different connections according to samples avoiding the contradiction between useful and redundant information.

Comparison with State-of-the-art. In this section, we evaluate our detector on COCO $test-dev$ set and compare with other state-of-the-art object detection approaches. For a fair comparison, we re-implement the corresponding baselines equipped with FPN on mmdetection. Besides, we use 2x training scheme without any bells and whistles to train our method on Faster R-CNN and Mask R-CNN. All results are shown in Table 3. It is obvious that our DSIC boosts the baselines by a significant improvement when integrated with one-stage and two-stage detectors. Also, our method outperforms the state-of-the-art detectors based on the same backbone without any bells and whistles. These improvements demonstrate the superior performance of our proposed framework.

4.3. Ablation Studies

In this section, we conduct the ablation experiments on $val2017$, using Faster R-CNN with FPN based on ResNet-50.

Ablation studies on each component. In order to verify the importance of our components in DSIC, we apply the ISG and CSG to the model gradually. We report the overall ablation studies in Table 4. "CSG" means our method abandons the FPN when compared with the baseline. Both of them have significant improvements and the performance of combination is much better which can prove the effectiveness of association.

Ablation studies on Gate operator. We further consider the mode selection in gate operator of two selection gate modules. Experimental results with different modes in two modules are shown in Table 5. In ISG, the three modes achieve the similar result which Softmax and Sigmoid ignore the noise in redundant information, which Tanh has a better performance. We consider that the information in the same stage is more complementary instead of mutual exclusion. However, the performance of three modes has obvious differences in CSG because of spatial contradiction of different levels. The Softmax operator takes account of four levels' data flow together ignoring the independence and inconsistency. Sigmoid operator considers the data flow individually which can't eliminate the redundant information. Tanh operator avoids the above defects and achieves the best result.

Table 5. Comparison with different mode selection of two gate modules.

Module	Mode selection	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ISG	Softmax	37.4	58.9	**40.7**	21.6	41.2	48.2
	Sigmoid	37.5	58.7	40.4	21.7	**41.3**	47.9
	Tanh	**37.6**	**59.0**	40.5	**21.7**	40.9	**49.0**
CSG	Softmax	37.4	59.0	40.4	21.3	41.0	48.4
	Sigmoid	37.6	**59.4**	40.4	**21.9**	41.4	48.8
	Tanh	**37.8**	59.1	**41.0**	21.8	**41.8**	**48.9**

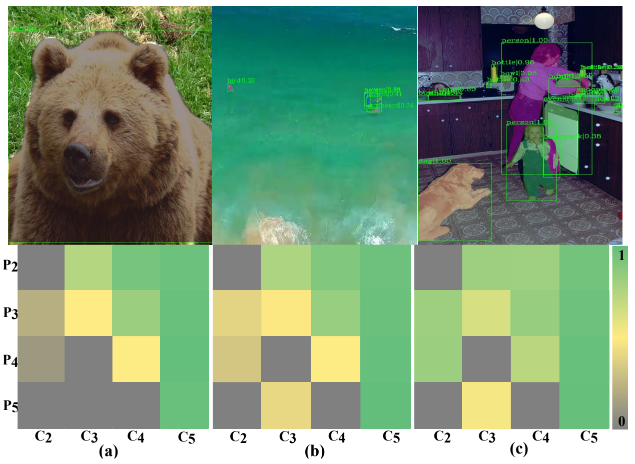

Fig. 3. The visual results of CSG. The first row is the detection results and the second row is the corresponding state matrix of each data flow path. (a) is the large scale input. (b) is the small scale input. And (c) has various scales of input.

4.4. Visualization

We present the visual results of CSG. AS shown in Fig 3, three different samples tend to select different connections according to the scale of objects, which validates the sample-individualized data flow path selection of DSIC. In addition, the regression task in detection need more high-resolution information whether large or small scale samples. But we find that the highest-resolution information of output is from the other three levels instead of itself which contains huge noise. By contrast, the classification task need more semantic information in high levels while the highest level output is from itself. Thus, DSIC dynamically selects the inputs and data flow paths of feature integration which reconciles the differences between the classification and regression to some extent.

5. CONCLUSION

In this paper, we have proposed a novel Dynamic Sample-Individualized Connector (DSIC) for multi-scale object detection, which dynamically adjusts network connections to fit different samples. With the help of two simple yet effective components, i.e., ISG and CSG, DSIC has shown the gen-

Table 3. Comparisons with the state-of-the-art methods on COCO $test-dev$. The symbol * means our re-implemented results.

Method	Backbone	Schedule	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
YoLOv2[18]	DarkNet-19	-	21.6	44.0	19.2	5.0	22.4	35.5
SSD512[4]	ResNet-101	-	31.2	50.4	33.3	10.2	34.5	49.8
Faster R-CNN[3]	ResNet-101-FPN	-	36.2	59.1	39.0	18.2	39.0	48.2
Deformable R-FCN[19]	Inception-ResNet-v2	-	37.5	58.0	40.8	19.4	40.1	52.5
Mask R-CNN[20]	ResNet-101-FPN	-	38.2	60.3	41.7	20.1	41.1	50.2
Libra R-CNN[11]	ResNet-101-FPN	1x	40.3	61.3	43.9	22.9	43.1	51.0
RetinaNet*	ResNet-50-FPN	1x	35.8	55.6	38.4	19.8	38.8	45.0
FCOS*	ResNet-101-FPN	2x	40.9	60.2	44.1	24.7	45.0	52.3
Faster R-CNN*	ResNet-101-FPN	2x	39.5	61.2	43.0	22.0	43.1	50.1
Mask R-CNN*	ResNet-101-FPN	2x	40.8	62.1	44.6	22.8	43.9	52.0
DSIC RetinaNet(ours)	ResNet-50-FPN	1x	37.2	57.3	39.8	20.3	40.0	47.1
DSIC FCOS(ours)	ResNet-101	2x	41.6	60.8	44.8	25.4	45.5	53.5
DSIC Faster R-CNN(ours)	ResNet-101	2x	41.0	62.4	44.6	25.0	44.7	52.6
DSIC Mask R-CNN(ours)	ResNet-101	2x	**42.6**	**63.0**	**46.6**	**25.5**	**45.8**	**52.8**

erality and effectiveness for both two-stage and single-stage detectors. Compared with previous approaches, DSIC offers several advantages: (1) instead of utilizing the special manual design and insufficient interactions, the whole process is dynamic and sample-individualized; (2) it performs better on multi-level feature integration and can be simply generalized in different computer vision tasks. The experiments on MS-COCO show the superiority of DSIC.

6. ACKNOWLEDGMENTS

This work was supported by the National Key Research and Development Program of China (Grant No. 2020AAA0106800, No. 2018AAA0102802, No. 2018AAA0102803, and No. 2018AAA0102800), the Natural Science Foundation of China (Grant No.61902401, No.61972071, No.61906052, No.62036011, No.61772225,and No.61721004), the CAS Key Research Program of Frontier Sciences (Grant No.QYZDJ-SSWJSC040), and the National Natural Science Foundation of Guangdong (No.2018B030311046). The work of Bing Li was also supported by the Youth Innovation Promotion Association, CAS.

7. REFERENCES

[1] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[2] Ross Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[3] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[4] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[5] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali 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.

[6] Zhaowei Cai and Nuno Vasconcelos, "Cascade r-cnn: Delving into high quality object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6154–6162.

[7] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollr, and C. Lawrence Zitnick, "Microsoft coco: Common objects in context," in *European Conference on Computer Vision*, 2014, pp. 740–755.

[8] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[9] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia, "Path aggregation network for instance segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8759–8768.

[10] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang, "Deep high-resolution representation learning for human pose estimation," *arXiv preprint arXiv:1902.09212*, 2019.

[11] Jiangmiao Pang, Kai Chen, Jianping Shi, Huajun Feng, Wanli Ouyang, and Dahua Lin, "Libra r-cnn: Towards balanced learning for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 821–830.

[12] Kai Chen, Yuhang Cao, Chen Change Loy, Dahua Lin, and Christoph Feichtenhofer, "Feature pyramid grids," *arXiv preprint arXiv:2004.03580*, 2020.

[13] Jin Chen, Xijun Wang, Zichao Guo, Xiangyu Zhang, and Jian Sun, "Dynamic region-aware convolution," *arXiv preprint arXiv:2003.12243*, 2020.

[14] Hongkai Zhang, Hong Chang, Bingpeng Ma, Naiyan Wang, and Xilin Chen, "Dynamic r-cnn: Towards high quality object detection via dynamic training," *arXiv preprint arXiv:2004.06002*, 2020.

[15] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer, "Automatic differentiation in pytorch," 2017.

[16] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al., "Mmdetection: Open mmlab detection toolbox and benchmark," *arXiv preprint arXiv:1906.07155*, 2019.

[17] Dong Zhang, Hanwang Zhang, Jinhui Tang, Meng Wang, Xiansheng Hua, and Qianru Sun, "Feature pyramid transformer.," *arXiv: Computer Vision and Pattern Recognition*, 2020.

[18] Joseph Redmon and Ali Farhadi, "Yolo9000: Better, faster, stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6517–6525.

[19] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun, "R-fcn: Object detection via region-based fully convolutional networks," in *Advances in neural information processing systems*, 2016, pp. 379–387.

[20] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.