Medical Image Segmentation

Deep Snake for Real-Time Instance Segmentation

CVPR 2020

Deep Snake for Real-Time Instance Segmentation

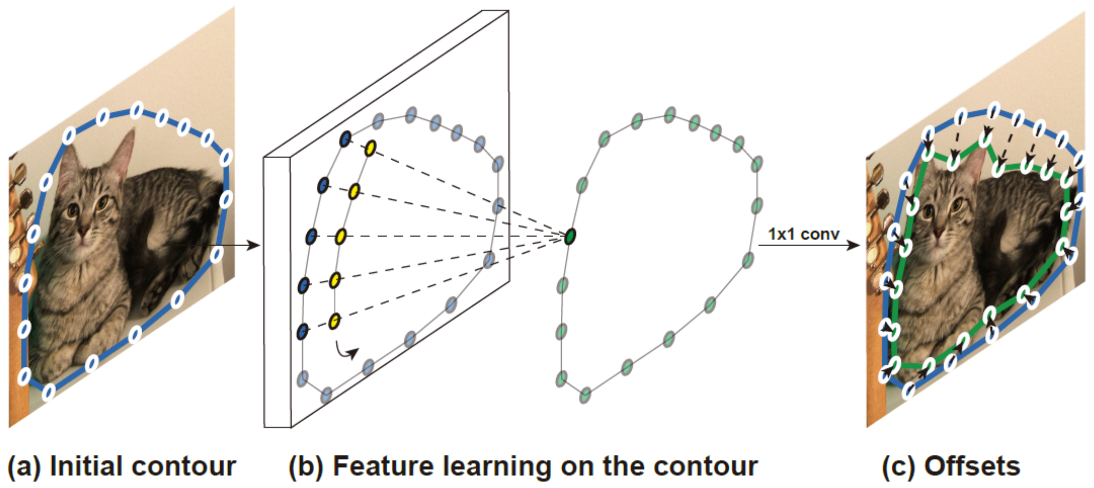


Figure 1. **The basic idea of deep snake.** Given an initial contour, image features are extracted at each vertex (a). Since the contour is a cycle graph, circular convolution is applied for feature learning on the contour (b). The blue, yellow and green nodes denote the input features, the kernel of circular convolution, and the output features, respectively. Finally, offsets are regressed at each vertex to deform the contour to match the object boundary (c).

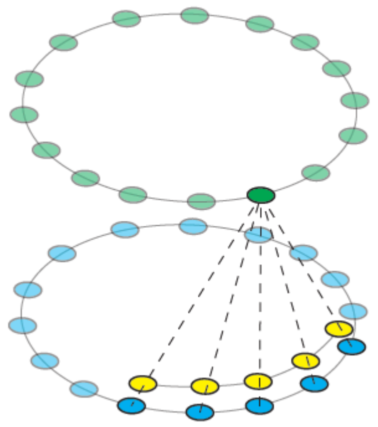
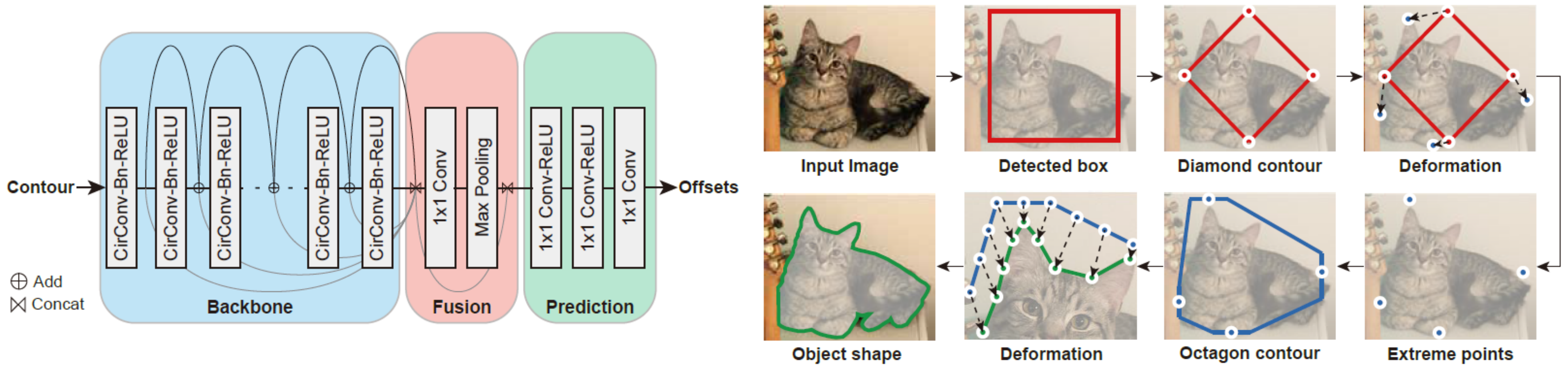


Figure 2. **Circular Convolution.** The blue nodes are the input features defined on a contour, the yellow nodes represent the kernel function, and the green nodes are the output features. The highlighted green node is the inner product between the kernel function and the highlighted blue nodes, which is the same as the standard convolution. The output features of circular convolution have the same length as the input features.

$$(f_N)_i \triangleq \sum_{j=-\infty}^{\infty} f_{i-jN}, \tag{1}$$

$$(f_N * k)_i = \sum_{j=-r}^{r} (f_N)_{i+j} k_j, \tag{2}$$



(a) Deep snake (b) Pipeline for instance segmentation

Figure 3. **Proposed contour-based model for instance segmentation.** (a) Deep snake consists of three parts: a backbone, a fusion block, and a prediction head. It takes a contour as input and outputs vertex-wise offsets to deform the contour. (b) Based on deep snake, we propose a two-stage pipeline for instance segmentation: initial contour proposal and contour deformation. The box proposed by the detector gives a diamond contour, whose four vertices are then shifted to object extreme points by deep snake. An octagon is constructed based on the extreme points. Taking the octagon as the initial contour, deep snake iteratively deforms it to match the object boundary.

	AP_{vol}	AP_{50}	AP_{70}
Baseline	50.9	58.8	43.5
+ Architecture	52.3	59.7	46.0
+ Initial proposal	53.6	61.1	47.6
+ Circular convolution	**54.4**	**62.1**	**48.3**

Table 1. **Ablation studies on SBD val set .** The baseline is a direct combination of Curve-gcn [25] and CenterNet [44]. The second model reserves the graph convolution and replaces the network architecture with our proposed one, which yields 1.4 AP_{vol} improvement. Then we add the initial contour proposal before contour deformation, which improves AP_{vol} by 1.3. The fourth row shows that replacing graph convolution with circular convolution further yields 0.8 AP_{vol} improvement.

	Iter. 1	Iter. 2	Iter. 3	Iter. 4	Iter. 5
Graph conv	50.2	51.5	53.6	52.2	51.6
Circular conv	50.6	54.2	**54.4**	54.0	53.2

Table 2. **Results of models with different convolution operators and different iterations** on SBD in terms of the AP_{vol} metric. Circular convolution outperforms graph convolution across all inference iterations. Furthermore, circular convolution with two iterations outperforms graph convolution with three iterations by 0.6 AP, indicating a stronger deforming ability. We also find that adding more iterations does not necessarily improve the performance, which shows that it might be harder to train the network with more iterations.

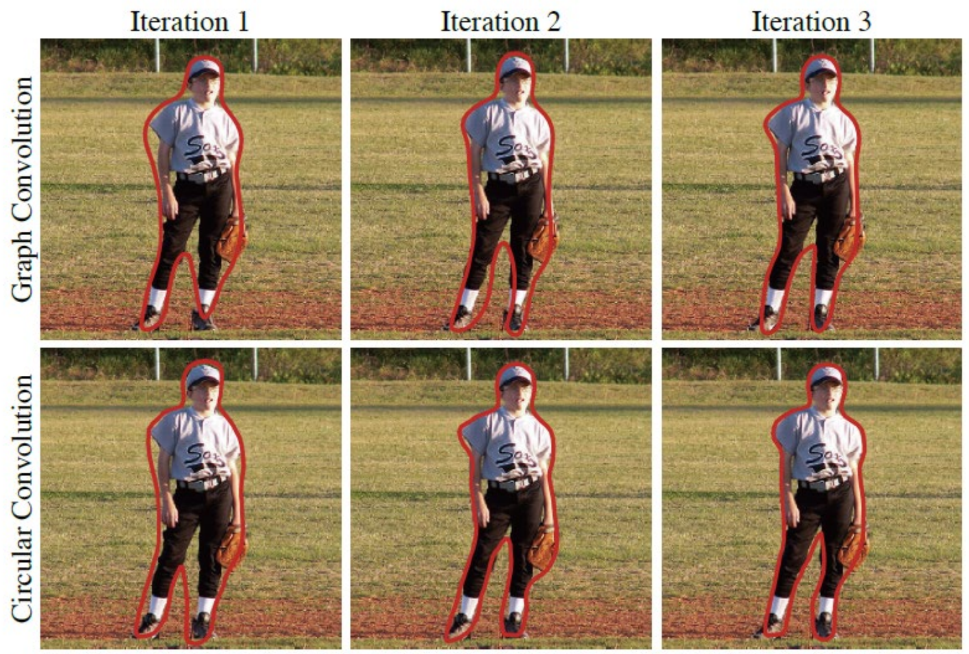


Figure 5. **Comparison between graph convolution (top) and circular convolution (bottom) on SBD.** The result of circular convolution with two iterations is visually better than that of graph convolution with three iterations.

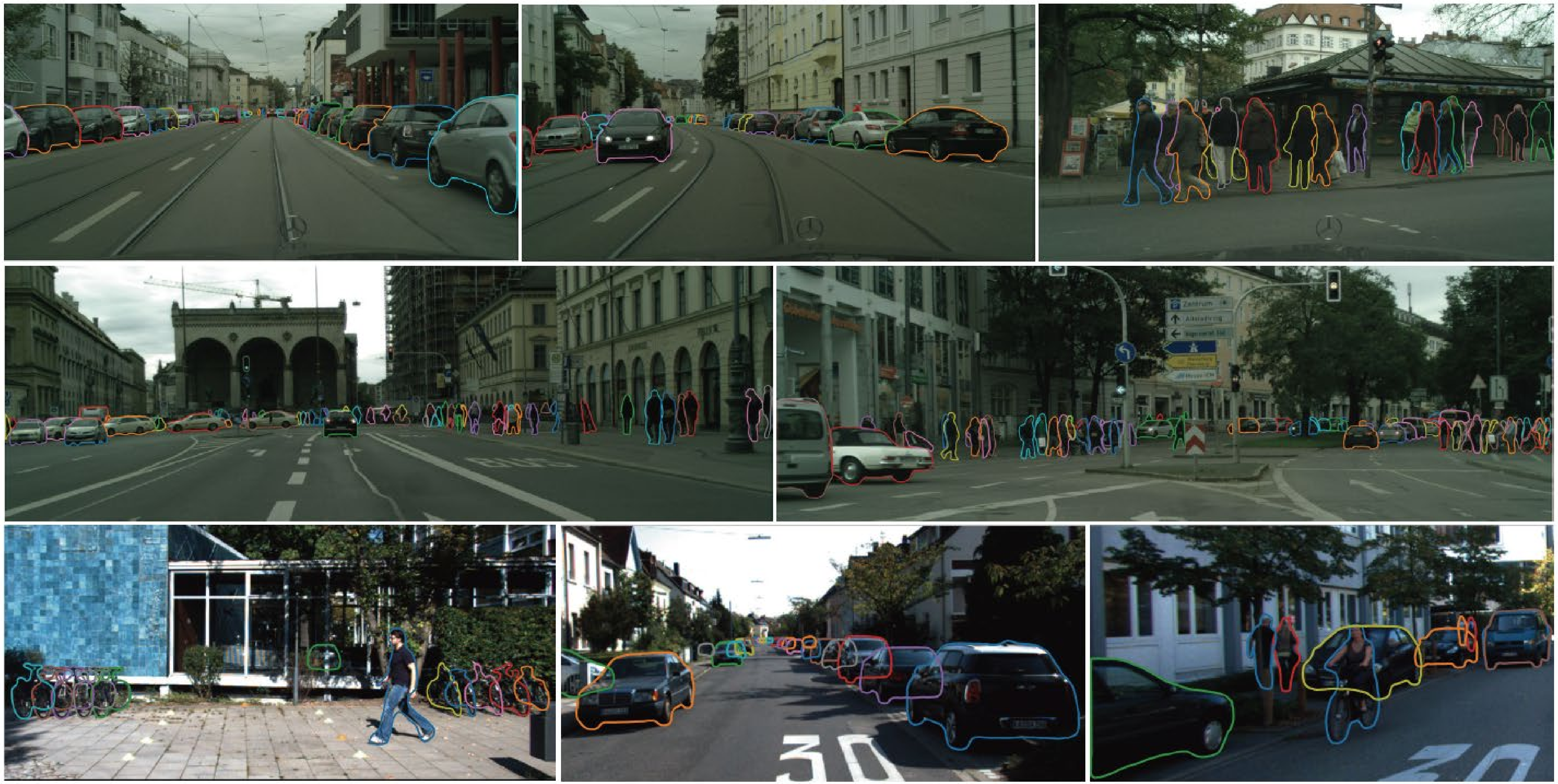


Figure 6. **Qualitative results on Cityscapes test and KINS test sets.** The first two rows show the results on Cityscapes, and the last row lists the results on KINS. Note that the results on KINS are for amodal instance segmentation.

ResNeSt: Split-Attention Networks

CVPR 2020

ResNeSt: Split-Attention Networks

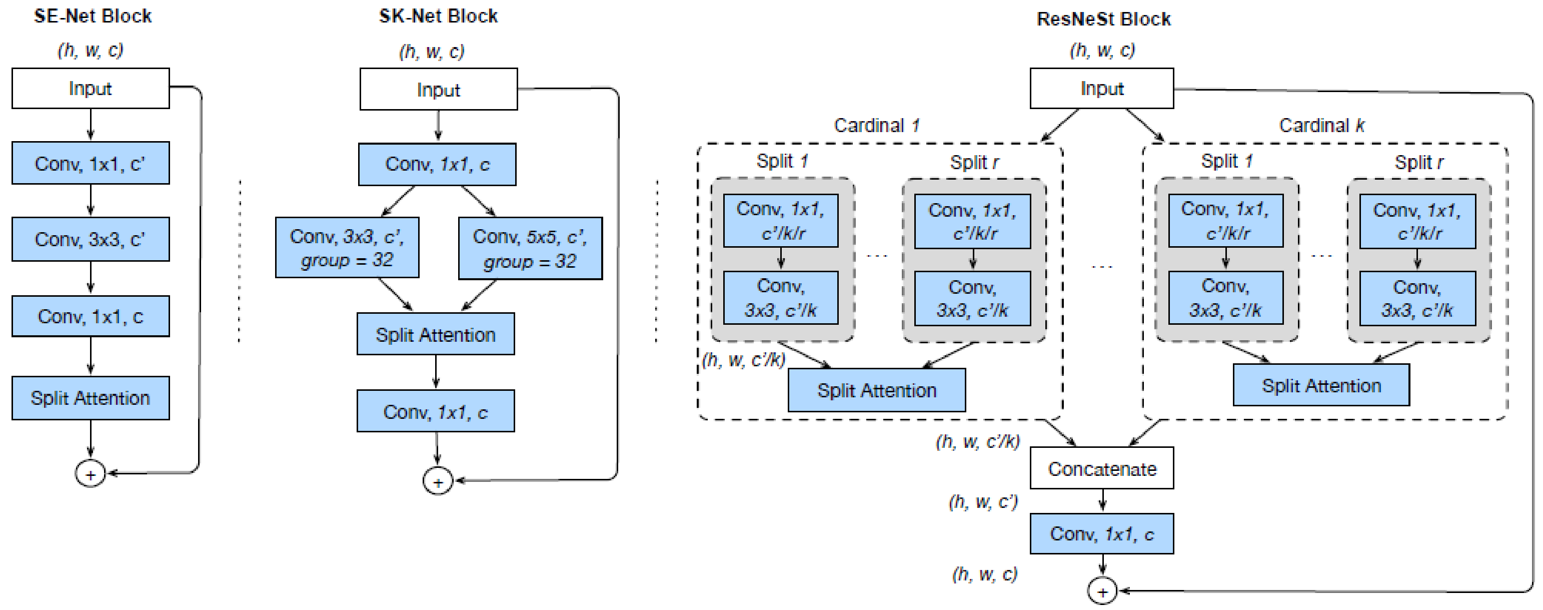


Fig. 1: Comparing our ResNeSt block with SE-Net [30] and SK-Net [38]. A detailed view of Split-Attention unit is shown in Figure 2. For simplicity, we show ResNeSt block in cardinality-major view (the featuremap groups with same cardinal group index reside next to each other). We use radix-major in the real implementation, which can be modularized and accelerated by group convolution and standard CNN layers (see supplementary material).

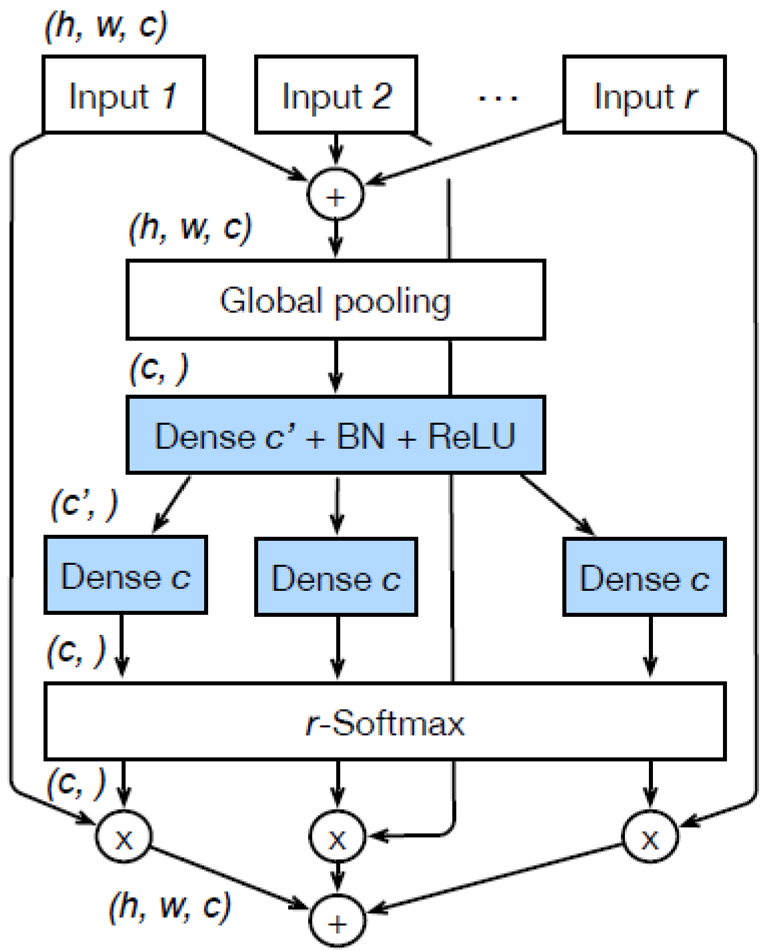
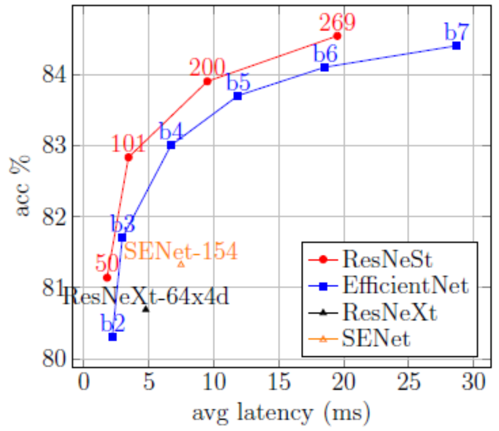


Fig. 2: Split-Attention within a cardinal group. For easy visualization in the figure, we use $c = C/K$ in this figure.



	Crop	#P	Acc%
ResNeSt-50 (ours)	224	27.5M	81.1
ResNeSt-101 (ours)	256	48.3M	82.8
ResNeSt-200 (ours)	320	70.2M	83.9
ResNeSt-269 (ours)	416	111M	84.5

	Backbone	#Params	Score%
FasterRCNN [46]	ResNet-50 [57]	34.9M	39.25
	ResNeSt-50 (ours)	36.8M	42.33
DeeplabV3 [7]	ResNet-50 [57]	42.2M	42.10
	ResNeSt-50 (ours)	44.0M	45.12

Table 1: (Left) Accuracy and latency trade-off on GPU using official code implementation (details in Section 5). (Right-Top) Top-1 accuracy on ImageNet using ResNeSt. (Right-Bottom) Transfer learning results: object detection mAP on MS-COCO [42] and semantic segmentation mIoU on ADE20K [71].

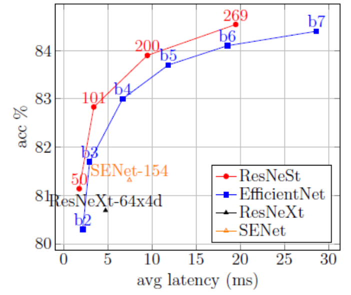
	#P	GFLOPs	top-1 acc (%) 224×	top-1 acc (%) 320×
ResNet-50 [23]	25.5M	4.14	76.15	76.86
ResNeXt-50 [60]	25.0M	4.24	77.77	78.95
SENet-50 [29]	27.7M	4.25	78.88	80.29
ResNetD-50 [26]	25.6M	4.34	79.15	79.70
SKNet-50 [38]	27.5M	4.47	79.21	80.68
ResNeSt-50-fast(ours)	27.5M	4.34	**80.64**	**81.43**
ResNeSt-50(ours)	27.5M	5.39	81.13	81.82
ResNet-101 [23]	44.5M	7.87	77.37	78.17
ResNeXt-101 [60]	44.3M	7.99	78.89	80.14
SENet-101 [29]	49.2M	8.00	79.42	81.39
ResNetD-101 [26]	44.6M	8.06	80.54	81.26
SKNet-101 [38]	48.9M	8.46	79.81	81.60
ResNeSt-101-fast(ours)	48.2M	8.07	**81.97**	**82.76**
ResNeSt-101(ours)	48.3M	10.2	82.27	83.00

Table 3: Image classification results on ImageNet, comparing our proposed ResNeSt with other ResNet variants of similar complexity in 50-layer and 101-layer configurations. We report top-1 accuracy using crop sizes 224 and 320.

	#P	GFLOPs	acc(%)
ResNetD-50 [26]	25.6M	4.34	78.31
+ mixup	25.6M	4.34	79.15
+ autoaug	25.6M	4.34	79.41
ResNeSt-50-fast	27.5M	4.34	80.64
ResNeSt-50	27.5M	5.39	81.13

Variant	#P	GFLOPs	img/sec	acc(%)
0s1x64d	25.6M	4.34	688.2	79.41
1s1x64d	26.3M	4.34	617.6	80.35
2s1x64d	27.5M	4.34	533.0	80.64
4s1x64d	31.9M	4.35	458.3	80.90
2s2x40d	26.9M	4.38	481.8	81.00

Table 2: Ablation study for ImageNet image classification. (Left) breakdown of improvements. (Right) *radix vs. cardinality* under ResNeSt-fast setting. For example *2s2x40d* denotes radix=2, cardinality=2 and width=40. Note that even radix=1 does not degrade any existing approach (see Equation 3).



	#P	crop	img/sec	acc(%)
ResNeSt-101(ours)	48M	256	**291.3**	**83.0**
EfficientNet-B4 [55]	19M	380	149.3	83.0
SENet-154 [29]	146M	320	133.8	82.7
NASNet-A [74]	89M	331	103.3	82.7
AmoebaNet-A [45]	87M	299	-	82.8
ResNeSt-200 (ours)	70M	320	**105.3**	**83.9**
EfficientNet-B5 [55]	30M	456	84.3	83.7
AmoebaNet-C [45]	155M	299	-	83.5
ResNeSt-269 (ours)	111M	416	**51.2**	**84.5**
GPipe	557M	-	-	84.3
EfficientNet-B7 [55]	66M	600	34.9	84.4

Table 4: Accuracy vs. Latency for SoTA CNN models on ImageNet with large crop sizes. Our ResNeSt model displays the best trade-off (additional details/results in Appendix). EfficientNet variants b2-b7 are described in [55]. ResNeSt variants use a different number of layers listed in red. Average Inference latency is measured on a NVIDIA V100 GPU using the original code implementation of each model with a mini-batch of size 16.