ResNeSt: Split-Attention Networks

Hang Zhang, Chongruo Wu*, Zhongyue Zhang, Yi Zhu, Haibin Lin, Zhi Zhang, Yue Sun, Tong He, Jonas Mueller, R. Manmatha, Mu Li, and Alexander Smola

Amazon, University of California, Davis*
{hzaws,chongrwu,zhongyue,yzaws,haibilin,zhiz,ysunmzn, htong,jonasmue,manmatha,mli,smola}@amazon.com

Abstract. While image classification models have recently continued to advance, most downstream applications such as object detection and semantic segmentation still employ ResNet variants as the backbone network due to their simple and modular structure. We present a modular *Split-Attention* block that enables attention across feature-map groups. By stacking these Split-Attention blocks ResNet-style, we obtain a new ResNet variant which we call *ResNeSt*. Our network preserves the overall ResNet structure to be used in downstream tasks straightforwardly without introducing additional computational costs.

ResNeSt models outperform other networks with similar model complexities. For example, ResNeSt-50 achieves 81.13% top-1 accuracy on ImageNet using a single crop-size of 224 × 224, outperforming previous best ResNet variant by more than 1% accuracy. This improvement also helps downstream tasks including object detection, instance segmentation and semantic segmentation. For example, by simply replace the ResNet-50 backbone with ResNeSt-50, we improve the mAP of Faster-RCNN on MS-COCO from 39.3% to 42.3% and the mIoU for DeeplabV3 on ADE20K from 42.1% to 45.1%[1].

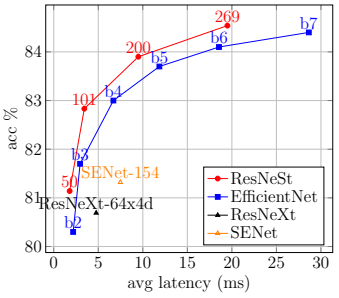
Keywords: ResNeSt, Image Classification, Transfer Learning, Object Detection, Semantic Segmentation, Instance Segmentation

1 Introduction

Image classification is a fundamental task in computer vision research. Networks trained for image classification often serve as the backbone of the neural networks designed for other applications, such as object detection [22, 46], semantic segmentation [6, 43, 73] and pose estimation [14, 58]. Recent work has significantly boosted image classification accuracy through large scale neural architecture search (NAS) [45, 55]. Despite their state-of-the-art performance, these NAS-derived models are usually not optimized for training efficiency or memory usage on general/commercial processing hardware (CPU/GPU) [36]. Due to excessive memory consumption, some of the larger versions of these models are

* Work done during an internship at Amazon.

[1] The source code is available at https://github.com/zhanghang1989/ResNeSt.



	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].

not even trainable on a GPU with an appropriate per-device batch-size[2] [55]. This has limited the adoption of NAS-derived models for other applications, especially tasks involving dense predictions such as segmentation.

Most recent work on downstream applications still uses the ResNet [23] or one of its variants as the backbone CNN. Its simple and modular design can be easily adapted to various tasks. However, since ResNet models are originally designed for image classification, they may not be suitable for various downstream applications because of the limited receptive-field size and lack of cross-channel interaction. This means that boosting performance on a given computer vision task requires "network surgery" to modify the ResNet to be more effective for that particular task. For example, some methods add a pyramid module [8, 69] or introduce long-range connections [56] or use cross-channel feature-map attention [15, 65]. While these approaches do improve the transfer learning performance for certain tasks, they raise the question: *Can we create a versatile backbone with universally improved feature representations, thereby improving performance across multiple tasks at the same time?* Cross-channel information has demonstrated success in downstream applications [56, 64, 65], while recent image classification networks have focused more on group or depth-wise convolution [27, 28, 54, 60]. Despite their superior computation and accuracy trade-off in classification tasks, these models do not transfer well to other tasks as their isolated representations cannot capture cross-channel relationships [27, 28]. Therefore, a network with cross-channel representations is desirable.

As the first contribution of this paper, we explore a simple architectural modification of the ResNet [23], incorporating feature-map split attention within the individual network blocks. More specifically, each of our blocks divides the feature-map into several groups (along the channel dimension) and finer-grained subgroups or splits, where the feature representation of each group is determined via a weighted combination of the representations of its splits (with weights cho-

[2] Note that the performance of batch normalization degrades for small batch-sizes as feature statistics can no longer be estimated reliably.

sen based on global contextual information). We refer to the resulting unit as a *Split-Attention* block, which remains simple and modular. By stacking several Split-Attention blocks, we create a ResNet-like network called *ResNeSt* (S stands for "split"). Our architecture requires no more computation than existing ResNet-variants, and is easy to be adopted as a backbone for other vision tasks.

The second contributions of this paper are large scale benchmarks on image classification and transfer learning applications. We find that models utilizing a ResNeSt backbone are able to achieve state of the art performance on several tasks, namely: image classification, object detection, instance segmentation and semantic segmentation. The proposed ResNeSt outperforms all existing ResNet variants and has the same computational efficiency and even achieves better speed-accuracy trade-offs than state-of-the-art CNN models produced via neural architecture search [55] as shown in Table 1. Our single Cascade-RCNN [3] model using a ResNeSt-101 backbone achieves 48.3% box mAP and 41.56% mask mAP on MS-COCO instance segmentation. Our single DeepLabV3 [7] model, again using a ResNeSt-101 backbone, achieves mIoU of 46.9% on the ADE20K scene parsing validation set, which surpasses the previous best result by more than 1% mIoU. Additional results can be found in Sections 5 and 6.

2 Related Work

Modern CNN Architectures. Since AlexNet [34], deep convolutional neural networks [35] have dominated image classification. With this trend, research has shifted from engineering handcrafted features to engineering network architectures. NIN [40] first uses a global average pooling layer to replace the heavy fully connected layers, and adopts 1×1 convolutional layers to learn non-linear combination of the featuremap channels, which is the first kind of featuremap attention mechanism. VGG-Net [47] proposes a modular network design strategy, stacking the same type of network blocks repeatedly, which simplifies the workflow of network design and transfer learning for downstream applications. Highway network [50] introduces highway connections which makes the information flow across several layers without attenuation and helps the network convergence. Built on the success of the pioneering work, ResNet [23] introduces an identity skip connection which alleviates the difficulty of vanishing gradient in deep neural network and allows network learning deeper feature representations. ResNet has become one of the most successful CNN architectures which has been adopted in various computer vision applications.

Multi-path and Feature-map Attention. Multi-path representation has shown success in GoogleNet [52], in which each network block consists of different convolutional kernels. ResNeXt [61] adopts group convolution [34] in the ResNet bottle block, which converts the multi-path structure into a unified operation. SE-Net [29] introduces a channel-attention mechanism by adaptively recalibrating the channel feature responses. SK-Net [38] brings the feature-map attention across two network branches. Inspired by the previous methods, our

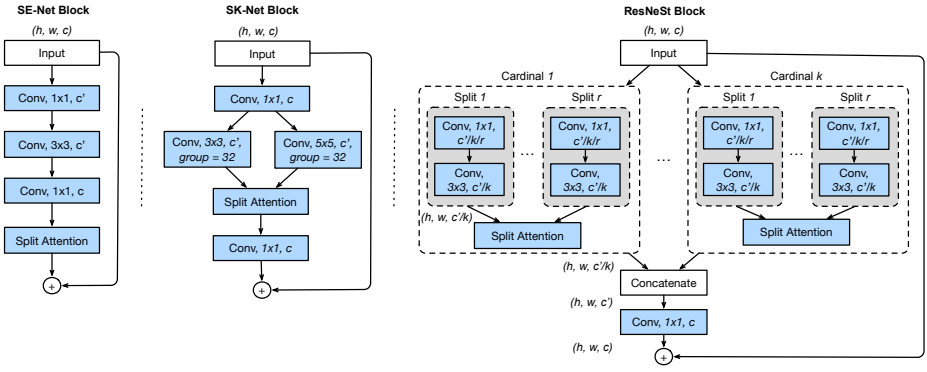


Fig. 1: Comparing our ResNeSt block with SE-Net [30] and SK-Net [38]. A detailed view of Split-Attention unit is shown in Figure 2. For simplicity, we show ResNeSt block in cardinality-major view (the featuremap groups with same cardinal group index reside next to each other). We use radix-major in the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

network generalizes the channel-wise attention into feature-map group representation, which can be modularized and accelerated using unified CNN operators.

Neural Architecture Search. With increasing computational power, interest has begun shifting from manually designed architectures to systematically searched architectures which are adaptively tailored to a particular task. Recent neural architecture search algorithms have adaptively produced CNN architectures that achieved state-of-the-art classification performance, such as: AmoebaNet [45], MNASNet [54], and EfficientNet [55]. Despite their great success in image classification, the meta network structures are distinct from each other, which makes it hard for downstream models to build upon. Instead, our model preserves ResNet meta structure, which can be directly applied on many existing downstream models [22, 41, 46, 69]. Our approach can also augment the search spaces for neural architecture search and potentially improve the overall performance, which can be studied in the future work.

3 Split-Attention Networks

We now introduce the Split-Attention block, which enables feature-map attention across different feature-map groups. Later, we describe our network instantiation and how to accelerate this architecture via standard CNN operators.

3.1 Split-Attention Block

Our *Split-Attention* block is a computational unit, consisting feature-map group and split attention operations. Figure 1 (Right) depicts an overview of a Split-Attention Block.

Feature-map Group. As in ResNeXt blocks [61], the feature can be divided into several groups, and the number of feature-map groups is given by a *cardinality* hyperparameter K. We refer to the resulting feature-map groups as *cardinal groups*. We introduce a new *radix* hyperparameter R that indicates the number of splits within a cardinal group, so the total number of feature groups is $G = KR$. We may apply a series of transformations $\{\mathcal{F}_1, \mathcal{F}_2, ...\mathcal{F}_G\}$ to each individual group, then the intermediate representation of each group is $U_i = \mathcal{F}_i(X)$, for $i \in \{1, 2, ...G\}$.

Split Attention in Cardinal Groups. Following [30, 38], a combined representation for each cardinal group can be obtained by fusing via an element-wise summation across multiple splits. The representation for k-th cardinal group is $\hat{U}^k = \sum_{j=R(k-1)+1}^{Rk} U_j$, where $\hat{U}^k \in \mathbb{R}^{H \times W \times C/K}$ for $k \in 1, 2, ...K$, and H, W and C are the block output feature-map sizes. Global contextual information with embedded channel-wise statistics can be gathered with global average pooling across spatial dimensions $s^k \in \mathbb{R}^{C/K}$ [29, 38]. Here the c-th component is calculated as:

$$s_c^k = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} \hat{U}_c^k(i, j). \tag{1}$$

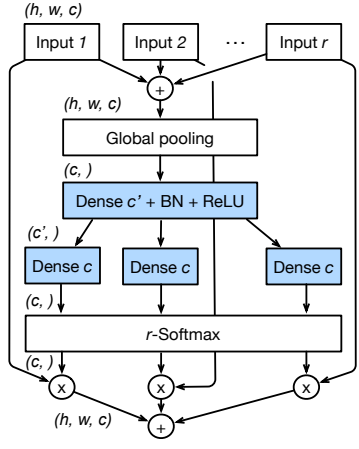


Fig. 2: Split-Attention within a cardinal group. For easy visualization in the figure, we use $c = C/K$ in this figure.

A weighted fusion of the cardinal group representation $V^k \in \mathbb{R}^{H \times W \times C/K}$ is aggregated using channel-wise soft attention, where each feature-map channel is produced using a weighted combination over splits. The c-th channel is calculated as:

$$V_c^k = \sum_{i=1}^{R} a_i^k(c) U_{R(k-1)+i}, \tag{2}$$

where $a_i^k(c)$ denotes a (soft) assignment weight given by:

$$a_i^k(c) = \begin{cases} \frac{exp(\mathcal{G}_i^c(s^k))}{\sum_{j=0}^{R} exp(\mathcal{G}_j^c(s^k))} & \text{if } R > 1, \\ \frac{1}{1+exp(-\mathcal{G}_i^c(s^k))} & \text{if } R = 1, \end{cases} \tag{3}$$

and mapping $\mathcal{G}_i^c$ determines the weight of each split for the c-th channel based on the global context representation s^k.

ResNeSt Block. The cardinal group representations are then concatenated along the channel dimension: $V = Concat\{V^1, V^2, ...V^K\}$. As in standard residual blocks, the final output Y of our Split-Attention block is produced using a shortcut connection: $Y = V + X$, if the input and output feature-map share the same shape. For blocks with a stride, an appropriate transformation $\mathcal{T}$ is applied to the shortcut connection to align the output shapes: $Y = V + \mathcal{T}(X)$. For example, $\mathcal{T}$ can be strided convolution or combined convolution-with-pooling.

Instantiation, Acceleration, and Computational Costs. Figure 1 (right) shows an instantiation of our Split-Attention block, in which the group transformation $\mathcal{F}_i$ is a 1×1 convolution followed by a 3×3 convolution, and the attention weight function $\mathcal{G}$ is parameterized using two fully connected layers with ReLU activation. We draw this figure in a cardinality-major view (the featuremap groups with same cardinality index reside next to each other) for easily describing the overall logic. By switching the layout to a radix-major view, this block can be easily accelerated using standard CNN layers (such as group convolution, group fully connected layer and softmax operation), which we will describe in details in the supplementary material. The number of parameters and FLOPS of a Split-Attention block are roughly the same as a residual block [23, 60] with the same cardinality and number of channels.

Relation to Existing Attention Methods. First introduced in SE-Net [29], the idea of squeeze-and-attention (called *excitation* in the original paper) is to employ a global context to predict channel-wise attention factors. With radix = 1, our Split-Attention block is applying a squeeze-and-attention operation to each cardinal group, while the SE-Net operates on top of the entire block regardless of multiple groups. Previous models like SK-Net [38] introduced feature attention between two network branches, but their operation is not optimized for training efficiency and scaling to large neural networks. Our method generalizes prior work on feature-map attention [29, 38] within a cardinal group setting [60], and its implementation remains computationally efficient. Figure 1 shows an overall comparison with SE-Net and SK-Net blocks.

4 Network and Training

We now describe the network design and training strategies used in our experiments. First, we detail a couple of tweaks that further improve performance, some of which have been empirically validated in [25].

4.1 Network Tweaks

Average Downsampling. When downstream applications of transfer learning are dense prediction tasks such as detection or segmentation, it becomes essential to preserve spatial information. Recent ResNet implementations usually apply the strided convolution at the 3×3 layer instead of the 1×1 layer to better

preserve such information [26,30]. Convolutional layers require handling feature-map boundaries with zero-padding strategies, which is often suboptimal when transferring to other dense prediction tasks. Instead of using strided convolution at the transitioning block (in which the spatial resolution is downsampled), we use an average pooling layer with a kernel size of 3×3 .

Tweaks from ResNet-D. We also adopt two simple yet effective ResNet modifications introduced by [26]: (1) The first 7×7 convolutional layer is replaced with three consecutive 3×3 convolutional layers, which have the same receptive field size with a similar computation cost as the original design. (2) A 2×2 average pooling layer is added to the shortcut connection prior to the 1×1 convolutional layer for the transitioning blocks with stride of two.

4.2 Training Strategy

Large Mini-batch Distributed Training. Following prior work [19,37], we train our models using 8 servers (64 GPUs in total) in parallel. Our learning rates are adjusted according to a cosine schedule [26,31]. We follow the common practice using linearly scaling-up the initial learning rate based on the mini-batch size. The initial learning rate is given by $\eta = \frac{B}{256}\eta_{base}$, where B is the mini-batch size and we use $\eta_{base} = 0.1$ as the base learning rate. This warm-up strategy is applied over the first 5 epochs, gradually increasing the learning rate linearly from 0 to the initial value for the cosine schedule [19, 39]. The batch normalization (BN) parameter γ is initialized to zero in the final BN operation of each block, as has been suggested for large batch training [19].

Label Smoothing Label smoothing was first used to improve the training of Inception-V2 [53]. Recall the cross entropy loss incurred by our network's predicted class probabilities q is computed against ground-truth p as:

$$\ell(p, q) = -\sum_{i=1}^{K} p_i \log q_i, \tag{4}$$

where K is total number of classes, p_i is the ground truth probability of the i-th class, and q_i is the network's predicted probability for the i-th class. As in standard image classification, we define: $q_i = \frac{exp(z_i)}{\sum_{j=1}^{K} exp(z_j)}$ where z_i are the logits produced by our network's ouput layer. When the provided labels are classes rather than class-probabilities (hard labels), $p_i = 1$ if i equals the ground truth class c, and is otherwise $= 0$. Thus in this setting: $\ell_{hard}(p, q) = -\log q_c = -z_c + \log(\sum_{j=1}^{K} exp(z_j))$. During the final phase of training, the logits z_j tend to be very small for $j \neq c$, while z_c is being pushed to its optimal value ∞, and this can induce overfitting [26, 53]. Rather than assigning hard labels as targets, label smoothing uses a smoothed ground truth probability:

$$p_i = \begin{cases} 1 - \varepsilon & \text{if } i = c, \\ \varepsilon/(K-1) & \text{otherwise} \end{cases} \tag{5}$$

with small constant $\varepsilon > 0$. This mitigates network overconfidence and overfitting.

Auto Augmentation. Auto-Augment [11] is a strategy that augments the training data with transformed images, where the transformations are learned adaptively. 16 different types of image jittering transformations are introduced, and from these, one augments the data based on 24 different combinations of two consecutive transformations such as shift, rotation, and color jittering. The magnitude of each transformation can be controlled with a relative parameter (e.g. rotation angle), and transformations may be probabilistically skipped. A search which tries various candidate augmentation policies returns the best 24 best combinations. One of these 24 policies is then randomly chosen and applied to each sample image during training. The original Auto-Augment implementation uses reinforcement learning to search over these hyperparameters, treating them as categorical values in a discrete search space. For continuous search spaces, it first discretizes the possible values before searching for the best.

Mixup Training. Mixup is another data augmentation strategy that generates a weighted combinations of random image pairs from the training data [67]. Given two images and their ground truth labels: $(x^{(i)}, y^{(i)}), (x^{(j)}, y^{(j)})$, a synthetic training example $(\hat{x}, \hat{y})$ is generated as:

$$\hat{x} = \lambda x^i + (1 - \lambda)x^j, \tag{6}$$

$$\hat{y} = \lambda y^i + (1 - \lambda)y^j, \tag{7}$$

where $\lambda \sim \text{Beta}(\alpha = 0.2)$ is independently sampled for each augmented example.

Large Crop Size. Image classification research typically compares the performance of different networks operating on images that share the same crop size. ResNet variants [23,26,29,60] usually use a fixed training crop size of 224, while the Inception-Net family [51–53] uses a training crop size of 299. Recently, the EfficientNet method [55] has demonstrated that increasing the input image size for a deeper and wider network may better trade off accuracy vs. FLOPS. For fair comparison, we use a crop size of 224 when comparing our ResNeSt with ResNet variants, and a crop size of 256 when comparing with other approaches.

Regularization. Very deep neural networks tend to overfit even for large datasets [68]. To prevent this, dropout regularization randomly masks out some neurons during training (but not during inference) to form an implicit network ensemble [29, 49, 68]. A dropout layer with the dropout probability of 0.2 is applied before the final fully-connected layer to the networks with more than 200 layers. We also apply DropBlock layers to the convolutional layers at the last two stages of the network. As a structured variant of dropout, DropBlock [18] randomly masks out local block regions, and is more effective than dropout for specifically regularizing convolutional layers.

Finally, we also apply weight decay (i.e. L2 regularization) which additionally helps stabilize training. Prior work on large mini-batch training suggests weight decay should only be applied to the weights of convolutional and fully connected layers [19,26]. We do not subject any of the other network parameters to weight decay, including bias units, γ and β in the batch normalization layers.

	#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).

5 Image Classification Results

Our first experiments study the image classification performance of ResNeSt on the ImageNet 2012 dataset [13] with 1.28M training images and 50K validation images (from 1000 different classes). As is standard, networks are trained on the training set and we report their top-1 accuracy on the validation set.

5.1 Implementation Details

We use data sharding for distributed training on ImageNet, evenly partitioning the data across GPUs. At each training iteration, a mini-batch of training data is sampled from the corresponding shard (without replacement). We apply the transformations from the learned Auto Augmentation policy to each individual image. Then we further apply standard transformations including: random size crop, random horizontal flip, color jittering, and changing the lighting. Finally, the image data are RGB-normalized via mean/standard-deviation rescaling. For mixup training, we simply mix each sample from the current mini-batch with its reversed order sample [26]. Batch Normalization [32] is used after each convolutional layer before ReLU activation [44]. Network weights are initialized using Kaiming Initialization [24]. A drop layer is inserted before the final classification layer with dropout ratio = 0.2. Training is done for 270 epochs with a weight decay of 0.0001 and momentum of 0.9, using a cosine learning rate schedule with the first 5 epochs reserved for warm-up. We use a mini-batch of size 8192 for ResNeSt-50, 4096 for ResNeSt 101, and 2048 for ResNeSt-{200, 269}. For evaluation, we first resize each image to 1/0.875 of the crop size along the short edge and apply a center crop. Our code implementation for ImageNet training uses GluonCV [21] with MXNet [9].

5.2 Ablation Study

ResNeSt is based on the ResNet-D model [26]. Mixup training improves the accuracy of ResNetD-50 from 78.31% to 79.15%. Auto augmentation further improves the accuracy by 0.26%. When employing our Split-Attention block to form a *ResNeSt-50-fast* model, accuracy is further boosted to 80.64%. In this

	#P	GFLOPs	top-1 acc (%)	
			224×	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

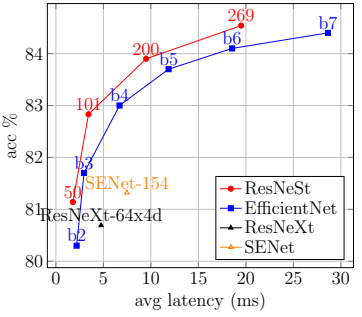
Table 3: Image classification results on ImageNet, comparing our proposed ResNeSt with other ResNet variants of similar complexity in 50-layer and 101-layer configurations. We report top-1 accuracy using crop sizes 224 and 320.

ResNeSt-fast setting, the effective average downsampling is applied prior to the 3×3 convolution to avoid introducing extra computational costs in the model. With the downsampling operation moved after the convolutional layer, ResNeSt-50 achieves 81.13% accuracy.

Radix vs. Cardinality. We conduct an ablation study on ResNeSt-variants with different radix/cardinality. In each variant, we adjust the network's width appropriately so that its overall computational cost remains similar to the ResNet variants. The results are shown in Table 2, where s denotes the radix, x the cardinality, and d the network width ($0s$ represents the use of a standard residual block as in ResNet-D [26]). We empirically find that increasing the radix from 0 to 4 continuously improves the top-1 accuracy, while also increasing latency and memory usage. Although we expect further accuracy improvements with even greater radix/cardinality, we employ Split-Attention with the $2s1x64d$ setting in subsequent experiments, to ensure these blocks scale to deeper networks with a good trade-off between speed, accuracy and memory usage.

5.3 Comparing against the State-of-the-Art

ResNet Variants. For comparison with ResNet variants [23, 26, 29, 38, 60], all networks (including ResNeSt) are trained using a crop size size of 224×224, and then evaluated using center crop with sizes 224×224 as well as 320×320. Following prior practice, we consider 50-layer and 101-layer networks in this benchmark. The use of average pooling instead of strided convolution as the down-sampling strategy increases computation by an extra 1 GFLOPS. For fair comparison with matched computational costs, we move the average pooling



	#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.

operation before the 3×3 convolutional layer to build a *ResNeSt-fast* model, where the convolutional layer operates on a downsampled feature-map. We use $2s1x64d$ (see Table 2) as the ResNeSt setting as it has better training and inference speed and less memory usage. Table 3 shows that our proposed ResNeSt outperforms all ResNet variants with a similar number of network parameters and FLOPS, including: ResNet [23], ResNeXt [60], SENet [29], ResNet-D [26] and SKNet [38]. Remarkably, our ResNeSt-50 achieves 80.64 top-1 accuracy, which is the first 50-layer ResNet variant that surpasses 80% on ImageNet.

Other CNN Models. To compare with CNN models trained using different crop size settings, we increase the training crop size for deeper models. We use a crop size of 256×256 for ResNeSt-200 and 320×320 for ResNeSt-269. Bicubic upsampling strategy is employed for input-size greater than 256. The results are shown in Table 4, where we compare the inference speed in addition to the number of parameters. We find that despite its advantage in parameters with accuracy trade-off, the widely used depth-wise convolution is not optimized for inference speed. In this benchmark, all inference speeds are measured using a mini-batch of 16 using the implementation [1] from the original author on a single NVIDIA V100 GPU. The proposed ResNeSt has better accuracy and latency trade-off than models found via neural architecture search.

6 Transfer Learning Results

6.1 Object Detection

We report our detection result on MS-COCO [42] in Table 10. All models are trained on COCO-2017 training set with 118k images, and evaluated on COCO-

	Method	Backbone	mAP%
Prior Work	Faster-RCNN [46]	ResNet101 [22]	37.3
		ResNeXt101 [5, 60]	40.1
		SE-ResNet101 [29]	41.9
	Faster-RCNN+DCN [12]	ResNet101 [5]	42.1
	Cascade-RCNN [2]	ResNet101	42.8
Our Results	Faster-RCNN [46]	ResNet50 [57]	39.25
		ResNet101 [57]	41.37
		ResNeSt50 (ours)	42.33
		ResNeSt101 (ours)	**44.72**
	Cascade-RCNN [2]	ResNet50 [57]	42.52
		ResNet101 [57]	44.03
		ResNeSt50 (ours)	45.41
		ResNeSt101 (ours)	**47.50**
	Cascade-RCNN [2]	ResNeSt200 (ours)	49.03

Table 5: Object detection results on the MS-COCO validation set. Both Faster-RCNN and Cascade-RCNN are significantly improved by our ResNeSt backbone.

2017 validation set with 5k images (aka. minival) using the standard COCO AP metric of single scale. We train all models with FPN [41], synchronized batch normalization [65] and image scale augmentation (short size of a image is picked randomly from 640 to 800). 1x learning rate schedule is used. We conduct Faster-RCNNs and Cascade-RCNNs experiments using Detectron2 [57]. For comparison, we simply replaced the vanilla ResNet backbones with our ResNeSt, while using the default settings for the hyper-parameters and detection heads [20, 57].

Compared to the baselines using standard ResNet, Our backbone is able to boost mean average precision by around 3% on both Faster-RCNNs and Cascade-RCNNs. The result demonstrates our backbone has good generalization ability and can be easily transferred to the downstream task. Notably, our ResNeSt50 outperforms ResNet101 on both Faster-RCNN and Cascade-RCNN detection models, using significantly fewer parameters. Detailed results in Table 10. We evaluate our Cascade-RCNN with ResNeSt101 deformable, that is trained using 1x learning rate schedule on COCO test-dev set as well. It yields a box mAP of 49.2 using single scale inference.

6.2 Instance Segmentation

To explore the generalization ability of our novel backbone, we also apply it to instance segmentation tasks. Besides the bounding box and category probability, instance segmentation also predicts object masks, for which a more accurate dense image representation is desirable.

We evaluate the Mask-RCNN [22] and Cascade-Mask-RCNN [2] models with ResNeSt-50 and ResNeSt-101 as their backbones. All models are trained along with FPN [41] and synchronized batch normalization. For data augmentation, input images' shorter side are randomly scaled to one of (640, 672, 704, 736, 768, 800). To fairly compare it with other methods, 1x learning rate schedule policy is applied, and other hyper-parameters remain the same. We re-train the baseline with the same setting described above, but with the standard ResNet. All our

	Method	Backbone	box mAP%	mask mAP%
Prior Work	DCV-V2 [72]	ResNet50	42.7	37.0
	HTC [4]	ResNet50	43.2	38.0
	Mask-RCNN [22]	ResNet101 [5]	39.9	36.1
	Cascade-RCNN [3]	ResNet101	44.8	38.0
Our Results	Mask-RCNN [22]	ResNet50 [57]	39.97	36.05
		ResNet101 [57]	41.78	37.51
		ResNeSt50 (ours)	42.81	38.14
		ResNeSt101 (ours)	**45.75**	**40.65**
	Cascade-RCNN [2]	ResNet50 [57]	43.06	37.19
		ResNet101 [57]	44.79	38.52
		ResNeSt50 (ours)	46.19	39.55
		ResNeSt101 (ours)	**48.30**	**41.56**

Table 6: Instance Segmentation results on the MS-COCO validation set. Both Mask-RCNN and Cascade-RCNN models are improved by our ResNeSt backbone. Models with our ResNeSt-101 outperform all prior work using ResNet-101.

experiments are trained on COCO-2017 dataset and using Detectron2 [57]. For the baseline experiments, the backbone we used by default is the MSRA version of ResNet, having stride-2 on the 1x1 conv layer. Both bounding box and mask mAP are reported on COCO-2017 validation dataset.

As shown in Table 6, our new backbone achieves better performance. For Mask-RCNN, ResNeSt50 outperforms the baseline with a gain of 2.85%/2.09% for box/mask performance, and ResNeSt101 exhibits even better improvement of 4.03%/3.14%. For Cascade-Mask-RCNN, the gains produced by switching to ResNeSt50 or ResNeSt101 are 3.13%/2.36% or 3.51%/3.04%, respectively. This suggests a model will be better if it consists of more Split-Attention modules. As observed in the detection results, the mAP of our ResNeSt50 exceeds the result of the standard ResNet101 backbone, which indicates a higher capacity of the small model with our proposed module. Finally, we also train a Cascade-Mask-RCNN with ResNeSt101-deformable using a 1x learning rate schedule. We evaluate it on the COCO test-dev set, yielding 50.0 box mAP, and 43.1 mask mAP respectively. Additional experiments under different settings are included in the supplementary material.

6.3 Semantic Segmentation

In transfer learning for the downstream task of semantic segmentation, we use the GluonCV [21] implementation of DeepLabV3 [7] as a baseline approach. Here a dilated network strategy [6,62] is applied to the backbone network, resulting in a stride-8 model. Synchronized Batch Normalization [65] is used during training, along with a polynomial-like learning rate schedule (with initial learning rate = 0.1). For evaluation, the network prediction logits are upsampled 8 times to calculate the per-pixel cross entropy loss against the ground truth labels. We use multi-scale evaluation with flipping [65, 69, 73].

We first consider the Cityscapes [10] dataset, which consists of 5K high-quality labeled images. We train each model on 2,975 images from the training set and report its mIoU on 500 validation images. Following prior work, we only consider 19 object/stuff categories in this benchmark. We have not used any

	Method	Backbone	pixAcc%	mIoU%
Prior Work	UperNet [59]	ResNet101	81.01	42.66
	PSPNet [69]	ResNet101	81.39	43.29
	EncNet [65]	ResNet101	81.69	44.65
	CFNet [66]	ResNet101	81.57	44.89
	OCNet [63]	ResNet101	-	45.45
	ACNet [17]	ResNet101	81.96	45.90
Ours	DeeplabV3 [7]	ResNet50 [21]	80.39	42.1
		ResNet101 [21]	81.11	44.14
		ResNeSt-50 (ours)	81.17	45.12
		ResNeSt-101 (ours)	**82.07**	**46.91**

	Method	Backbone	mIoU%
Prior Work	DANet [16]	ResNet101	77.6
	PSANet [70]	ResNet101	77.9
	PSPNet [69]	ResNet101	78.4
	AAF [33]	ResNet101	79.2
	DeeplabV3 [7]	ResNet101	79.3
	OCNet [63]	ResNet101	80.1
Ours	DeeplabV3 [7]	ResNet50 [21]	78.72
		ResNet101 [21]	79.42
		ResNeSt-50 (ours)	79.87
		ResNeSt-101 (ours)	**80.42**

Table 7: Semantic segmentation results on validation set of: ADE20K (Left), Citscapes (Right). Models are trained without coarse labels or extra data.

coarse labeled images or any extra data in this benchmark. Our ResNeSt backbone boosts the mIoU achieved by DeepLabV3 models by around 1% while maintaining a similar overall model complexity. Notably, the DeepLabV3 model using our ResNeSt-50 backbone already achieves better performance than DeepLabV3 with a much larger ResNet-101 backbone.

ADE20K [71] is a large scene parsing dataset with 150 object and stuff classes containing 20K training, 2K validation, and 3K test images. All networks are trained on the training set for 120 epochs and evaluated on the validation set. Table 7 shows the resulting pixel accuracy (pixAcc) and mean intersection-of-union (mIoU). The performance of the DeepLabV3 models are dramatically improved by employing our ResNeSt backbone. Analogous to previous results, the DeepLabv3 model using our ResNeSt-50 backbone already outperforms DeepLabv3 using a deeper ResNet-101 backbone. DeepLabV3 with a ResNeSt-101 backbone achieves 82.07% pixAcc and 46.91% mIoU, which to our knowledge, is the best single model that has been presented for ADE20K.

7 Conclusion

This work proposed the ResNeSt architecture with a novel Split-Attention block that universally improves the learned feature representations to boost performance across image classification, object detection, instance segmentation and semantic segmentation. In the latter downstream tasks, the empirical improvement produced by simply switching the backbone network to our ResNeSt is substantially better than task-specific modifications applied to a standard backbone such as ResNet. Our Split-Attention block is easy to work with and computationally efficient, and thus should be broadly applicable across vision tasks.

References

1. Tensorflow Efficientnet. `https://github.com/tensorflow/tpu/tree/master/models/official/efficientnet`, accessed: 2020-03-04
2. Cai, Z., Vasconcelos, N.: Cascade r-cnn: Delving into high quality object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 6154–6162 (2018)

3. Cai, Z., Vasconcelos, N.: Cascade r-cnn: High quality object detection and instance segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence (2019)
4. Chen, K., Pang, J., Wang, J., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Shi, J., Ouyang, W., et al.: Hybrid task cascade for instance segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4974–4983 (2019)
5. Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., Zhang, Z., Cheng, D., Zhu, C., Cheng, T., Zhao, Q., Li, B., Lu, X., Zhu, R., Wu, Y., Dai, J., Wang, J., Shi, J., Ouyang, W., Loy, C.C., Lin, D.: MMDetection: Open mmlab detection toolbox and benchmark. arXiv preprint arXiv:1906.07155 (2019)
6. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. arXiv:1606.00915 (2016)
7. Chen, L.C., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587 (2017)
8. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. arXiv preprint arXiv:1802.02611 (2018)
9. Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C., Zhang, Z.: Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. arXiv preprint arXiv:1512.01274 (2015)
10. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3213–3223 (2016)
11. Cubuk, E.D., Zoph, B., Mane, D., Vasudevan, V., Le, Q.V.: Autoaugment: Learning augmentation strategies from data. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 113–123 (2019)
12. Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., Wei, Y.: Deformable convolutional networks. In: Proceedings of the IEEE international conference on computer vision. pp. 764–773 (2017)
13. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR09 (2009)
14. Fang, H.S., Xie, S., Tai, Y.W., Lu, C.: Rmpe: Regional multi-person pose estimation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2334–2343 (2017)
15. Fu, J., Liu, J., Tian, H., Fang, Z., Lu, H.: Dual attention network for scene segmentation. arXiv preprint arXiv:1809.02983 (2018)
16. Fu, J., Liu, J., Tian, H., Li, Y., Bao, Y., Fang, Z., Lu, H.: Dual Attention Network for Scene Segmentation (2019)
17. Fu, J., Liu, J., Wang, Y., Li, Y., Bao, Y., Tang, J., Lu, H.: Adaptive context network for scene parsing. In: Proceedings of the IEEE international conference on computer vision. pp. 6748–6757 (2019)
18. Ghiasi, G., Lin, T.Y., Le, Q.V.: Dropblock: A regularization method for convolutional networks. In: Advances in Neural Information Processing Systems. pp. 10727–10737 (2018)
19. Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., He, K.: Accurate, large minibatch sgd: Training imagenet in 1 hour. arXiv preprint arXiv:1706.02677 (2017)

20. Guo, J., He, H., He, T., Lausen, L., Li, M., Lin, H., Shi, X., Wang, C., Xie, J., Zha, S., Zhang, A., Zhang, H., Zhang, Z., Zhang, Z., Zheng, S., Zhu, Y.: Gluoncv and gluonnlp: Deep learning in computer vision and natural language processing. Journal of Machine Learning Research **21**(23), 1–7 (2020), http://jmlr.org/papers/v21/19-429.html
21. Guo, J., He, H., He, T., Lausen, L., Li, M., Lin, H., Shi, X., Wang, C., Xie, J., Zha, S., et al.: Gluoncv and gluonnlp: Deep learning in computer vision and natural language processing. Journal of Machine Learning Research **21**(23), 1–7 (2020)
22. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. arXiv preprint arXiv:1703.06870 (2017)
23. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385 (2015)
24. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision. pp. 1026–1034 (2015)
25. He, T., Zhang, Z., Zhang, H., Zhang, Z., Xie, J., Li, M.: Bag of tricks to train convolutional neural networks for image classification. arXiv preprint arXiv:1812.01187 (2018)
26. He, T., Zhang, Z., Zhang, H., Zhang, Z., Xie, J., Li, M.: Bag of tricks for image classification with convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 558–567 (2019)
27. Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al.: Searching for mobilenetv3. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1314–1324 (2019)
28. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
29. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. arXiv preprint arXiv:1709.01507 (2017)
30. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7132–7141 (2018)
31. Huang, G., Liu, Z., Weinberger, K.Q., van der Maaten, L.: Densely connected convolutional networks. arXiv preprint arXiv:1608.06993 (2016)
32. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning. pp. 448–456 (2015)
33. Ke, T.W., Hwang, J.J., Liu, Z., Yu, S.X.: Adaptive Affinity Fields for Semantic Segmentation. In: European Conference on Computer Vision (ECCV) (2018)
34. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
35. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
36. Lee, J., Won, T., Hong, K.: Compounding the performance improvements of assembled techniques in a convolutional neural network. arXiv preprint arXiv:2001.06268 (2020)
37. Li, M.: Scaling distributed machine learning with system and algorithm co-design. Ph.D. thesis, PhD thesis, Intel (2017)
38. Li, X., Wang, W., Hu, X., Yang, J.: Selective kernel networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 510–519 (2019)

39. Lin, H., Zhang, H., Ma, Y., He, T., Zhang, Z., Zha, S., Li, M.: Dynamic mini-batch sgd for elastic distributed training: Learning in the limbo of resources. arXiv preprint arXiv:1904.12043 (2019)
40. Lin, M., Chen, Q., Yan, S.: Network in network. arXiv preprint arXiv:1312.4400 (2013)
41. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2117–2125 (2017)
42. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European conference on computer vision. pp. 740–755. Springer (2014)
43. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3431–3440 (2015)
44. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th international conference on machine learning (ICML-10). pp. 807–814 (2010)
45. Real, E., Aggarwal, A., Huang, Y., Le, Q.V.: Regularized evolution for image classifier architecture search. In: Proceedings of the aaai conference on artificial intelligence. vol. 33, pp. 4780–4789 (2019)
46. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. pp. 91–99 (2015)
47. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
48. Singh, B., Najibi, M., Davis, L.S.: Sniper: Efficient multi-scale training. In: Advances in neural information processing systems. pp. 9310–9320 (2018)
49. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. Journal of machine learning research **15**(1), 1929–1958 (2014)
50. Srivastava, R.K., Greff, K., Schmidhuber, J.: Highway networks. arXiv preprint arXiv:1505.00387 (2015)
51. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.A.: Inception-v4, inception-resnet and the impact of residual connections on learning. In: Thirty-first AAAI conference on artificial intelligence (2017)
52. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1–9 (2015)
53. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2818–2826 (2016)
54. Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., Le, Q.V.: Mnasnet: Platform-aware neural architecture search for mobile. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2820–2828 (2019)
55. Tan, M., Le, Q.V.: Efficientnet: Rethinking model scaling for convolutional neural networks. arXiv preprint arXiv:1905.11946 (2019)
56. Wang, X., Girshick, R., Gupta, A., He, K.: Non-local neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7794–7803 (2018)

57. Wu, Y., Kirillov, A., Massa, F., Lo, W.Y., Girshick, R.: Detectron2. `https://github.com/facebookresearch/detectron2` (2019)
58. Xiao, B., Wu, H., Wei, Y.: Simple baselines for human pose estimation and tracking. In: Proceedings of the European conference on computer vision (ECCV). pp. 466–481 (2018)
59. Xiao, T., Liu, Y., Zhou, B., Jiang, Y., Sun, J.: Unified perceptual parsing for scene understanding. arXiv preprint arXiv:1807.10221 (2018)
60. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. arXiv preprint arXiv:1611.05431 (2016)
61. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1492–1500 (2017)
62. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions. arXiv preprint arXiv:1511.07122 (2015)
63. Yuan, Y., Chen, X., Wang, J.: Object-contextual representations for semantic segmentation. arXiv preprint arXiv:1909.11065 (2019)
64. Yuhui Yuan, J.W.: Ocnet: Object context network for scene parsing. arXiv preprint arXiv:1809.00916 (2018)
65. Zhang, H., Dana, K., Shi, J., Zhang, Z., Wang, X., Tyagi, A., Agrawal, A.: Context encoding for semantic segmentation. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018)
66. Zhang, H., Zhang, H., Wang, C., Xie, J.: Co-occurrent features in semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 548–557 (2019)
67. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412 (2017)
68. Zhang, X., Li, Z., Change Loy, C., Lin, D.: Polynet: A pursuit of structural diversity in very deep networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 718–726 (2017)
69. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
70. Zhao, H., Zhang, Y., Liu, S., Shi, J., Loy, C.C., Lin, D., Jia, J.: PSANet: Point-wise Spatial Attention Network for Scene Parsing. In: European Conference on Computer Vision (ECCV) (2018)
71. Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., Torralba, A.: Scene parsing through ade20k dataset. In: Proc. CVPR (2017)
72. Zhu, X., Hu, H., Lin, S., Dai, J.: Deformable convnets v2: More deformable, better results. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9308–9316 (2019)
73. Zhu, Y., Sapra, K., Reda, F.A., Shih, K.J., Newsam, S., Tao, A., Catanzaro, B.: Improving Semantic Segmentation via Video Propagation and Label Relaxation (2019)
74. Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V.: Learning transferable architectures for scalable image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 8697–8710 (2018)

Appendix

A Radix-major Split-Attention Block

For easily visualizing the concept of Split-Attention, we employ cardinality-major implementation in the methods description of the main paper, where the groups with the same cardinal index reside next to each other physically. The cardinality-major implementation is straightforward and intuitive, but is difficult to modularize and accelerate using standard CNN operators. Therefore, we adopt the radix-major implementation in our experiments.

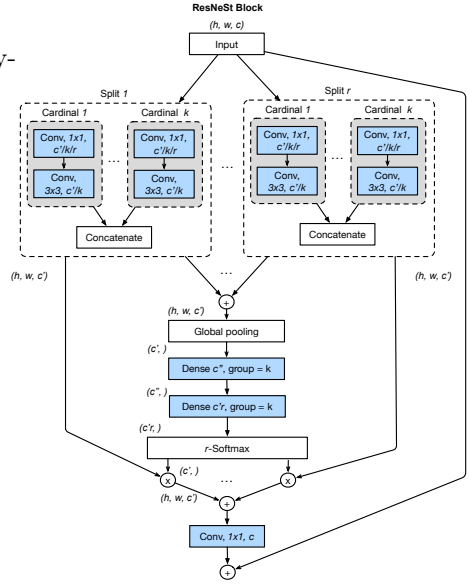


Fig. 3: Radix-major implementation of ResNeSt block, where the featuremap groups with same radix index but different cardinality are next to each other physically. This implementation can be easily accelerated, because the 1×1 convolutional layers can be unified into a layer and the 3×3 convolutional layers can be implemented using group convolution with the number of groups equal to RK.

Figure 3 gives an overview of the Split-Attention block in radix-major layout. The input feature-map is first divided into RK groups, in which each group has a cardinality-index and radix-index. In this layout, the groups with same radix-index reside next to each other in the memory. Then, we can conduct a summation across different splits, so that the feature-map groups with the same cardinality-index but different radix-index are fused together. This operation is identical to fuse across splits within each cardinal groups in the cardinality-major implementation described in the main paper. Similarly, a global pooling layer aggregates over the spatial dimension, while keeps the channel dimension separated, which is the same as conducting global pooling to each individual cardinal groups then concatenate the results. Then two consecutive fully connected (FC) layers with number of groups equal to cardinality are added after pooling layer to predict the attention weights for each splits. The use of grouped FC layers makes it identical to apply each pair of FCs separately on top each cardinal groups.

With this implementation, the first 1×1 convolutional layers can be unified into one layer and the 3×3 convolutional layers can be implemented using a single grouped convolution with the number of groups of RK. Therefore, the Split-Attention block is modularized and implemented using standard CNN operators.

Method	Backbone	OKS AP% w/o flip	OKS AP% w/ flip
SimplePose [58]	ResNet50 [21]	71.0/91.2/78.6	72.2/92.2/79.9
	ResNet101 [21]	72.6/91.3/80.8	73.6/92.3/81.1
	ResNeSt50 (ours)	72.3/92.3/80.0	73.4/92.4/81.2
	ResNeSt101 (ours)	73.6/92.3/80.9	74.6/92.4/82.1

Table 8: Pose estimation results on MS-COCO dataset in terms of OKS AP.

	Method	**Backbone**	Deformable	**mAP%**
Prior Work	DCNv2 [72]	ResNet101 [23]	v2	44.8
		ResNeXt101 [60]	v2	45.3
	TridentNet [12]	ResNet101 [23]	v1	46.8
		ResNet101* [12]	v1	48.4
	SNIPER [48]	ResNet101* [12]	v1	46.1
	Cascade-RCNN [3]	ResNet101 [23]	n/a	42.8
	Cascade-RCNN [3]	ResNeSt101 (ours)	v2	**49.2**

Table 9: Object detection results on the MS-COCO test-dev set. The single model of Cascade-RCNN with ResNeSt backbone using deformable convolution [12] achieves 49.2% mAP, which outperforms all previous methods. (* means using multi-scale evaluation.)

B Additional Experiments

B.1 Pose Estimation

We investigate the effect of backbone on pose estimation task. The baseline model is SimplePose [58] with ResNet50 and ResNet101 implemented in GluonCV [21]. As comparison we replace the backbone with ResNeSt50 and ResNeSt101 respectively while keeping other settings unchanged. The input image size is fixed to 256x192 for all runs. We use Adam optimizer with batch size 32 and initial learning rate 0.001 with no weight decay. The learning rate is divided by 10 at the 90th and 120th epoch. The experiments are conducted on COCO Keypoints dataset, and we report the OKS AP for results without and with flip test. Flip test first makes prediction on both original and horizontally flipped images, and then averages the predicted keypoint coordinates as the final output.

From Table 8, we see that models backboned with ResNeSt50/ResNeSt101 significantly outperform their ResNet counterparts. Besides, with ResNeSt50 backbone the model achieves performance similar with ResNet101 backbone.

B.2 Object Detection and Instance Segmentation

For object detection, we add deformable convolution to our Cascade-RCNN model with ResNeSt-101 backbone and train the model on the MS-COCO training set for 1x schedule. The resulting model achieves 49.2% mAP on COCO test-dev set, which surpass all previous methods including these employing multi-scale evaluation. Detailed results are shown in Table 10.

We include more results of instance segmentation, shown in Table 11, from the models trained with 1x/3x learning rate schedules and with/without SyncBN.

	Method	Backbone	Deformable	box mAP%	mask mAP%
Prior Work	DCNv2 [72]	ResNet101 [23]	v2	45.8	39.7
		ResNeXt101 [60]	v2	46.7	40.5
	SNIPER [48]	ResNet101* [12]	v1	47.1	41.3
	Cascade-RCNN [3]	ResNext101 [60]	n/a	45.8	38.6
	Cascade-RCNN [3]	ResNeSt101 (ours)	v2	**50.0**	**43.0**

Table 10: Instance Segmentation results on the MS-COCO test-dev set. * denote multi-scale inference.

Method	lr schedule	SyncBN	Backbone	box mAP%	mask mAP%
Mask-RCNN [22]	1×		ResNet50 [57]	38.60	35.20
			ResNet101 [57]	40.79	36.93
			ResNeSt50 (ours)	40.85	36.99
			ResNeSt101 (ours)	**43.98**	**39.33**
		✓	ResNet50 [57]	39.97	36.05
			ResNet101 [57]	41.78	37.51
			ResNeSt50 (ours)	42.81	38.14
			ResNeSt101 (ours)	**45.75**	**40.65**
	3×		ResNet50 [57]	41.00	37.20
			ResNet101 [57]	42.90	38.60
			ResNeSt50 (ours)	43.32	38.91
			ResNeSt101 (ours)	**45.37**	**40.56**
Cascade-RCNN [2]	1×		ResNet50 [57]	42.10	36.40
			ResNet101 [57]	44.00	38.08
			ResNeSt50 (ours)	44.56	38.27
			ResNeSt101 (ours)	**46.86**	**40.23**
		✓	ResNet50 [57]	43.06	37.19
			ResNet101 [57]	44.79	38.52
			ResNeSt50 (ours)	46.19	39.55
			ResNeSt101 (ours)	**48.30**	**41.56**
	3×		ResNet50 [57]	44.3	38.5
			ResNet101 [57]	45.57	39.54
			ResNeSt50 (ours)	46.39	39.99
			ResNeSt101 (ours)	**47.70**	**41.16**

Table 11: Instance Segmentation results on the MS-COCO validation set. Comparing models trained w/ and w/o SyncBN, and using 1× and 3× learning rate schedules.

All of resutls are reported on COCO val dataset. For both 50/101-layer settings, our ResNeSt backbones still outperform the corresponding baselines with different lr schedules. Same as the Table. 6 in the main text, our ResNeSt50 also exceeds the result of the standard ResNet101.

We also evaluate our ResNeSt with and without deformable convolution v2 [72]. With its help, we are able to obtain a higher performance, shown in Table 12. It indicates our designed module is compatible with deformable convolution.

C Future Work and Conclusions

With Split-Attention block, we introduce a new hyperparameter radix to the ResNet series. We conduct a brief ablation study on a few combinations of radix, cardinality and width. However, a comprehensive study on the hyper-parameter combinations can further boost the performance of the ResNeSt model, especially

Method	Deformable [72] (v2)	box mAP%	mask mAP%
Cascade-RCNN [2]		48.30	41.56
	✓	**49.39**	**42.56**

Table 12: The results of Cascade-Mask-RCNN on COCO val set. The ResNeSt-101 is applied with and without deformable convolution v2 [72]. It shows that our split-attention module is compatible with other existing modules.

on specific applications. One interesting topic is finding low latency models on different hardwares through neural architecture search.

Beyond the paper contributions, we empirically find several minor conclusions which may be helpful for peers:

- depth-wise convolution is not optimal for training and inference efficiency on GPU,
- model accuracy get saturated on ImageNet with a fixed input image size,
- increasing input image size can get better accuracy and FLOPS trade-off.
- bicubic upsampling strategy is needed for large crop-size (≥ 320).