ResNeSt: Split-Attention Networks

Hang Zhang[1], Chongruo Wu[2], Zhongyue Zhang[3], Yi Zhu[4], Haibin Lin[5], Zhi Zhang[4],
Yue Sun[6], Tong He[4], Jonas Mueller[4], R. Manmatha[4], Mu Li[4], Alexander Smola[4]

Meta[1], UC Davis[2], Snap[3], Amazon[4], ByteDance[5], SenseTime[6]

zhanghang@fb.com, crwu@ucdavis.edu, zzhang5@snapchat.com, haibin.lin@bytedance.com,
sunyue1@sensetime.com, {yzaws,zhiz,htong,jonasmue,manmatha,mli,smola}@amazon.com

Abstract

The ability to learn richer network representations generally boosts the performance of deep learning models. To improve representation-learning in convolutional neural networks, we present a multi-branch architecture, which applies channel-wise attention across different network branches to leverage the complementary strengths of both feature-map attention and multi-path representation. Our proposed Split-Attention module provides a simple and modular computation block that can serve as a drop-in replacement for the popular residual block, while producing more diverse representations via cross-feature interactions. Adding a Split-Attention module into the architecture design space of RegNet-Y and FBNetV2 directly improves the performance of the resulting network. Replacing residual blocks with our Split-Attention module, we further design a new variant of the ResNet model, named ResNeSt, which outperforms EfficientNet in terms of the accuracy/latency trade-off.

1. Introduction

Deep convolutional neural networks (CNNs) have become the fundamental approach for image classification and other transfer learning tasks in computer vision. As the key component of the CNNs, a convolutional layer learns a set of filters which aggregates the neighborhood information with spatial and channel connections. This operation is suitable to capture *correlated features* with the output channels densely connected to each input channel. Inception models [52, 53] explore the multi-path representation to learn *independent features*, where the input is split into a few lower dimensional embeddings, transformed by different sets of convolutional filters and then merged by concatenation. This strategy encourages the feature exploration by decoupling the input channel connections [60].

The neuron connections in visual cortex have inspired the development of CNNs in the past decades [29]. The main theme of visual representation learning is discovering salient features for a given task [72]. Prior work has modeled spatial and channel dependencies [2,26,42], and incorporated attention mechanism [26, 35, 56]. SE-like channel-wise attention [26] employs global pooling to squeeze the channel statistics, and predicts a set of attention factors to apply channel-wise multiplication with the original featuremaps. This mechanism models the interdependencies of featuremap channels, which uses the global context information to selectively highlight or de-emphasize the features [26, 35]. This attention mechanism is similar to attentional selection stage of human primary visual cortex [71], which finds the informative parts for recognizing objects. Human/animals perceive various visual patterns using the cortex in separate regions that respond to different and particular visual features [45]. This strategy makes it easy to identify subtle but dominant differences of similar objects in the neural perception system. Similarly, if we can build a CNN architecture to capture individual salient attributes for different visual features, we would improve the network representation for image classification.

In this paper, we present a simple architecture which combines the channel-wise attention strategy with multi-path network layout. Our method captures cross-channel feature correlations, while preserving independent representation in the meta structure. A module of our network performs a set of transformations on low dimensional embeddings and concatenates their outputs as in a multi-path network. Each transformation incorporates channel-wise attention strategy to capture interdependencies of the featuremap. We further simplify the architecture to make each transformation share the same topology (*e.g.* Fig 2 (Right)). We can parameterize the network architecture with only a few variables. In addition, such setting also allows us to accelerate the training using identical implementation with unified CNN operators. We refer to such computation block

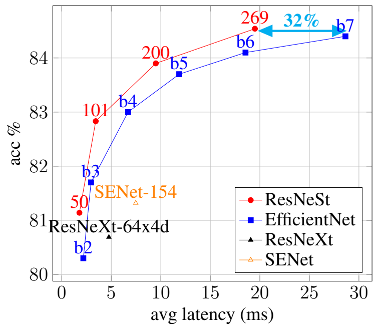

Figure 1. ResNeSt outperforms EfficientNet in accuracy-latency trade-offs on GPU. Notably, ResNeSt-269 has achieved better accuracy than EfficientNet-B7 with 32% less latency. (details in Section 4).

as *Split-Attention Block*. Stacking several Split-Attention blocks in ResNet style, we create a new ResNet variant which we refer to as *Split-Attention Network (ResNeSt)*.

We first benchmark the performance of the proposed Split-Attention Module in ResNet and mobile network settings. Split-Attention Module is added to RegNet-Y and FBNetV2 design spaces and the network performance got improved directly without bells and whistles. We then study the proposed ResNeSt. It achieves better speed-accuracy trade-offs on GPU than EfficientNet [54] produced via neural architecture search as shown in Table 4.

2. Related Work

CNN Architectures. Since AlexNet [32], deep convolutional neural networks [33] have dominated image classification. With this trend, research has shifted from engineering handcrafted features to engineering network architectures. NIN [37] 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 [48] proposes a modular network design strategy, stacking the same type of network blocks repeatedly, which simplifies both 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 [22] introduces an identity skip connection which alleviates the difficulty of vanishing gradient in deep neural network and allows network to learn improved feature representations. ResNet has become one of the most successful CNN architectures which has been adopted in various computer vision applications.

Multi-path and featuremap 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 [32] in the ResNet bottle block, which converts the multi-path structure into a unified operation. SE-Net [26] introduces a channel-attention mechanism by adaptively recalibrating the channel feature responses. Recently, SK-Net [35] brings the featuremap attention across two network branches. Inspired by the previous methods, our network integrates the channel-wise attention with multi-path network representation.

Neural Architecture Search. With increasing computational power, research interest has begun shifting from manually designed architectures to systematically searched architectures. Recent work explored efficient neural architecture search via parameter sharing [40, 43] and have achieved great success in low-latency and low-complexity CNN models [3, 57]. However, searching a large-scale neural network is still challenging due to the high GPU memory usage via parameter sharing. EfficientNet [54] first searches in a small setting and then scale up the network complexity systematically. Depthwise convolution is widely used in neural architecture search (NAS) work, due to better accuracy and FLOPs trade-off. However, Radosavovic *et al.* [44] uses a statistical method to analyze the network design space, and empirically find that fewer FLOPs does not necessarily indicate lower network latency. Furthermore, NAS approach doesn't deepen our understanding of the architecture [44], the quality of the resulting network still relies on the manually designed search spaces. Our work augments the search spaces for neural architecture search and potentially improve the performance, which can be studied in the future work.

3. Split-Attention Block

Split-Attention block is a computational unit, consisting of *featuremap group* and *split attention* operations. Figure 2 (Right) depicts an overview of a Split-Attention Block.

Featuremap Group. As in ResNeXt blocks [61], the feature can be divided into several groups, and the number of featuremap groups is given by a *cardinality* hyperparameter K. We refer to the resulting featuremap groups as *cardinal groups*. In this paper, 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 [27, 35], 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

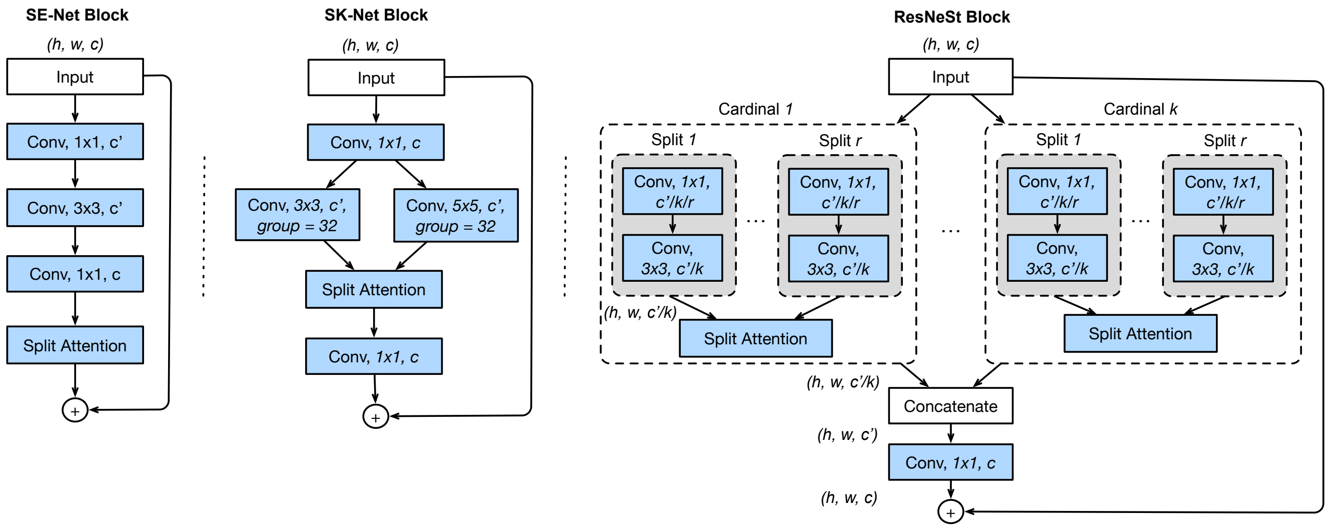

Figure 2. Comparing our ResNeSt block with SE-Net [27] and SK-Net [35]. A detailed view of Split-Attention unit is shown in Figure 3. For simplicity, we show ResNeSt block in cardinality-major view (the featuremap groups with same cardinal group index reside next to each other). We use radix-major in the real implementation, which can be modularized and accelerated by group convolution and standard CNN layers (see supplementary material).

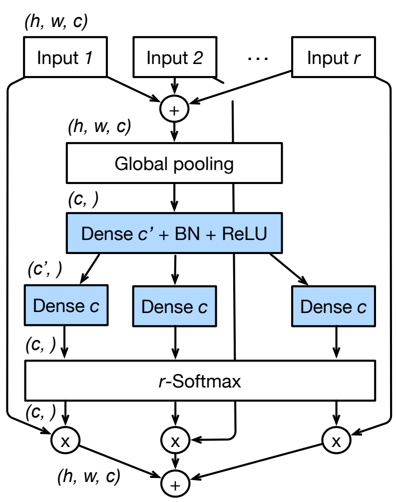

Figure 3. Split-Attention within a cardinal group. For easy visualization in the figure, we use $c = C/K$ in this figure.

$k \in 1, 2, ...K$, and H, W and C are the block output featuremap sizes. Global contextual information with embedded channel-wise statistics can be gathered with global average pooling across spatial dimensions $s^k \in \mathbb{R}^{C/K}$ [26, 35]. Here the c-th component is calculated as:

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

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 featuremap channel is produced using a weighted combination over splits. Then the c-th channel is calculated as:

$$V^k_c = \sum_{i=1}^{R} a^k_i(c) U_{R(k-1)+i}, \tag{2}$$

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

$$a^k_i(c) = \begin{cases} \frac{exp(\mathcal{G}^c_i(s^k))}{\sum_{j=1}^{R} exp(\mathcal{G}^c_j(s^k))} & \text{if } R > 1, \\ \frac{1}{1+exp(-\mathcal{G}^c_i(s^k))} & \text{if } R = 1, \end{cases} \tag{3}$$

and mapping $\mathcal{G}^c_i$ 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 featuremap 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 and Computational Costs. Figure 2 (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. The number of parameters and FLOPS of a Split-Attention block are roughly the same as a standard residual block [22, 60] with the same cardinality and number of channels.

Relation to Existing Attention Methods. First introduced in SE-Net [26], 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. SK-Net [35] introduces feature attention between two network streams. Setting radix $= 2$, the Split-Attention block applies SK-like attention to each cardinal group. Our method generalizes prior work of featuremap attention [26, 35] within a cardinal group setting [60], and its implementation remains computationally efficient. Figure 2 shows an overall comparison with SE-Net and SK-Net blocks.

4. Experiments on Split-Attention Module

In order to fully understand the proposed Split-Attention module, we add Split-Attention module directly into existing network design spaces and compare the performance. We consider both ResNet and Mobile setting in this benchmark.

4.1. ResNet Setting

Implementation Details. We adopt standard training recipe on ImageNet in this benchmark [22,48]. Specifically, we randomly resize the image along the shorter edge with a resulting size of [256, 480] while keeping the aspect ratio. The resulting images are randomly flipped horizontally and normalized by subtracting per-pixel mean and divided by standard deviation. Standard color augmentation is used as in [48]. We initialize the weights using MSRA init [23]. We use SGD with a mini-batch size of 256. The starting learning rate is set to 0.1 and cosine learning rate decay is used as in [28]. The models are trained for 120 epochs. We use a weight-decay of 0.0001 and momentum of 0.9. No other strategies or tricks are used in this section, and all models are using exactly the same setting.

ResNet [22] setting has become a gold-stand benchmark for studying CNN modules. RegNet [44] conducts a systematic search for the featuremap groups in ResNet variants and achieves superior performance over EfficientNet [54]. We add the proposed split-attention module into RegNet design space and we refer to the resulting variant as *RegNet-SA* (Split-Attention). For simplicity, we only add $split = 1$ to the design space in this section, which results in a small design space[1]. This helps us understand how much "multi-head SE" module can outperform standard SE module [26]. We add extra requirement of the network to preserve 50-layer ResNet meta architecture, *i.e.* each stage has [3, 4, 6, 3] bottleneck blocks. We randomly sample 50 different network configurations and report the best configuration in Table 1. Our model RegNet-SA-50 outperform the RegNet-Y [44] with similar network complexity. We also retrain the manually designed SENet-50 [26] and SKNet-50 [35] for reference.

[1]We have also studied $split = 2$ and observed similar performance. Results will be included in the Appendix.

4.2. Mobile Setting

In addition to ResNet setting, we also study the proposed Split-Attention module in mobile network setting. Recently, Differential Neural Architecture Search (DNAS) work has achieved great progress in pushing the SoTA results of mobile networks. We follow the prior work of FBNetV2 [55], but use Split-Attention module instead of SE module in the MBConv block with the number of splits in [1,2,4]. We refer to the FBNetV2 with Split-Attention module as *SA-FBNetV2*. We find that adding Split-Attention module directly boosts the accuracy and FLOPs trade-off of FBNetV2. Results are shown in Table 2. Similar results are also reported in an independent work of FP-NAS [62].

Our first experiments study the image classification performance of ResNeSt on the ImageNet 2012 dataset [14] 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. ResNeSt Implementation

With the proposed Split-Attention module, we stack the network block in ResNet style, resulting in a ResNet variant which we refer to as *Split Attention Networks (ResNeSt)*. First, we detail a couple of tweaks that further improve the results, some of which have been empirically validated in [24]. Then, we describe the advanced training strategies we use to further boosting the network performance.

5.1. Network Tweaks

Average Downsampling. For transfer learning on 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 previous 1×1 layer to better preserve such information [25, 27]. Convolutional layers require handling featuremap 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 [25]: (1) The first 7×7 convolutional layer is replaced with three consecutive 3×3 convolutional layers, which have the same

Stage	RegNet-Y-4GF	RegNet-SA-50 (ours)	SENet-50	SKNet-50
C1	3×3, 32, stride 2		7×7, 64, stride 2 3×3 max pool, stride 2	
C2	$\begin{bmatrix} 128 \\ 128, G=2 \\ \text{SE} \\ 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 216 \\ 216, G=27 \\ \text{SA}[r1, c6] \\ 216 \end{bmatrix} \times 3$	$\begin{bmatrix} 128 \\ 128, G=32 \\ 256 \\ \text{SE} \end{bmatrix} \times 3$	$\begin{bmatrix} 128 \\ 128, G=32 \\ \text{SK} \\ 256 \end{bmatrix} \times 3$
C3	$\begin{bmatrix} 192 \\ 192, G=3 \\ \text{SE} \\ 192 \end{bmatrix} \times 6$	$\begin{bmatrix} 288 \\ 288, G=18 \\ \text{SA}[r1, c24] \\ 288 \end{bmatrix} \times 4$	$\begin{bmatrix} 256 \\ 256, G=32 \\ 512 \\ \text{SE} \end{bmatrix} \times 4$	$\begin{bmatrix} 256 \\ 256, G=32 \\ \text{SK} \\ 512 \end{bmatrix} \times 6$
C4	$\begin{bmatrix} 512 \\ 512, G=8 \\ \text{SE} \\ 512 \end{bmatrix} \times 12$	$\begin{bmatrix} 720 \\ 720, G=45 \\ \text{SA}[r1, c18] \\ 720 \end{bmatrix} \times 6$	$\begin{bmatrix} 512 \\ 512, G=32 \\ 1024 \\ \text{SE} \end{bmatrix} \times 6$	$\begin{bmatrix} 512 \\ 512, G=32 \\ \text{SK} \\ 1024 \end{bmatrix} \times 6$
C5	$\begin{bmatrix} 1088 \\ 1088, G=17 \\ \text{SE} \\ 1088 \end{bmatrix} \times 2$	$\begin{bmatrix} 1344 \\ 1344, G=42 \\ \text{SA}[r1, c28] \\ 1344 \end{bmatrix} \times 3$	$\begin{bmatrix} 1024 \\ 1024, G=32 \\ 2048 \\ \text{SE} \end{bmatrix} \times 3$	$\begin{bmatrix} 1024 \\ 1024, G=32 \\ \text{SK} \\ 2048 \end{bmatrix} \times 3$
FC	7×7 global average pool, 1000-d fc, softmax			
#P	21.05M	21.67M	27.7M	27.5M
FLOPs	4.00G	4.04G	4.25G	4.47G
Acc	79.3	**79.6**	79.1	78.9

Table 1. Adding Split-Attention module into the RegNet design space, the resulting model RegNet-SA-50 outperforms RegNet-Y with similar FLOPs and number of parameters, even with extra constrain of preserving 50-layer ResNet style. We also retrain the manually designed SENet-50 & SKNet models for reference. All models are trained using the same recipe.

Model	FLOPs	Acc
FBNetV2-F1	56M	68.3
SA-FBNetV2-F1(ours)	65M	69.9
FBNetV2-F3	126M	73.2
SA-FBNetV2-F3(ours)	136M	74.1
FBNetV2-F4	238M	76.0
SA-FBNetV2-F4(ours)	262M	76.6

Table 2. Replacing the SE module with Split-Attention module in FBNetV2 directly boots the network accuracy and FLOPs trade-off.

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.

5.2. Training Strategy

Large Mini-batch Distributed Training.[2] For effectively training deep CNN models, we follow the prior work [18, 34, 36] to train our models using 8 servers (64 GPUs in total) in parallel. Our learning rates are adjusted according to a cosine schedule [25, 28]. 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 [18, 36]. 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 [18].

Label Smoothing. We adapt label smoothing as in prior work [25, 53] to avoid overfitting. Soft-label is used:

$$p_i = \begin{cases} 1 - \varepsilon & \text{if } i = c, \\ \varepsilon/(K-1) & \text{otherwise} \end{cases} \tag{4}$$

where c is the ground-truth class, K is number of classes, $\varepsilon > 0$ is a small constant. This mitigates network overconfidence and overfitting.

Auto Augmentation. Auto-Augment [12] 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.

Mixup Training. Mixup is another data augmentation strategy that generates a weighted combinations of random image pairs from the training data [65]. 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:

$$\hat{x} = \lambda x^i + (1-\lambda)x^j, \tag{5}$$

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

[2] Note that large mini-batch training only improve the training speed but not the accuracy. Instead, it often degrades the results.

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 [22, 25, 26, 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 [54] 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 [26, 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 [17] randomly masks out local block regions, and is more effective than dropout for specifically regularizing convolutional layers.

6. Experiments on ResNeSt

6.1. 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 [25]. Batch Normalization [30] is used after each convolutional layer before ReLU activation [41]. Network weights are initialized using Kaiming Initialization [23]. 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 [19] with MXNet [10].

6.2. Ablation Study

ResNeSt is based on the ResNet-D model [25]. 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 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 3, 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 [25]). 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.

6.3. Comparing against the State-of-the-Art

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.

7. Transfer Learning Results

7.1. Object Detection

We report our detection result on MS-COCO [39] in Table 5. All models are trained on COCO-2017 training set with 118k images, and evaluated on COCO-2017 validation set with 5k images (aka. minival) using the standard COCO AP metric of single scale. We train all models with

	#P	GFLOPs	acc(%)
ResNetD-50 [25]	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 3. 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 [54]	19M	380	149.3	83.0
SENet-154 [26]	146M	320	133.8	82.7
NASNet-A [76]	89M	331	103.3	82.7
AmoebaNet-A [46]	87M	299	-	82.8
ResNeSt-200 (ours)	70M	320	**105.3**	**83.9**
EfficientNet-B5 [54]	30M	456	84.3	83.7
AmoebaNet-C [46]	155M	299	-	83.5
ResNeSt-269 (ours)	111M	416	**51.2**	**84.5**
GPipe	557M	-	-	84.3
EfficientNet-B7 [54]	66M	600	34.9	84.4

Table 4. Accuracy vs. Throughput for SoTA CNN models on ImageNet. Our ResNeSt model displays the best trade-off. 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.

	Method	Backbone	mAP%
Prior Work	Faster-RCNN [47]	ResNet101 [21]	37.3
		ResNeXt101 [7, 60]	40.1
		SE-ResNet101 [26]	41.9
	Faster-RCNN+DCN [13]	ResNet101 [7]	42.1
	Cascade-RCNN [4]	ResNet101	42.8
Our Results	Faster-RCNN [47]	ResNet50 [58]	39.25
		ResNet101 [58]	41.37
		ResNeSt50 (ours)	42.33
		ResNeSt101 (ours)	**44.72**
	Cascade-RCNN [4]	ResNet50 [58]	42.52
		ResNet101 [58]	44.03
		ResNeSt50 (ours)	45.41
		ResNeSt101 (ours)	**47.50**
	Cascade-RCNN [4]	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.

FPN [38], synchronized batch normalization [66] 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 [58]. 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, 58].

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 5. 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.

7.2. 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 [21] and Cascade-Mask-RCNN [4] models with ResNeSt-50 and ResNeSt-101 as their backbones. All models are trained along with FPN [38] 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 experiments are trained on COCO-2017 dataset and using Detectron2 [58]. 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

	Method	Backbone	box mAP%	mask mAP%
Prior Work	DCV-V2 [74]	ResNet50	42.7	37.0
	HTC [6]	ResNet50	43.2	38.0
	Mask-RCNN [21]	ResNet101 [7]	39.9	36.1
	Cascade-RCNN [5]	ResNet101	44.8	38.0
Our Results	Mask-RCNN [21]	ResNet50 [58]	39.97	36.05
		ResNet101 [58]	41.78	37.51
		ResNeSt50 (ours)	42.81	38.14
		ResNeSt101 (ours)	**45.75**	**40.65**
	Cascade-RCNN [4]	ResNet50 [58]	43.06	37.19
		ResNet101 [58]	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.

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.

7.3. Semantic Segmentation

In transfer learning for semantic segmentation, we use the GluonCV [19] implementation of DeepLabV3 [9] as a baseline approach. Here a dilated network strategy [8, 63] is applied to the backbone network, resulting in a stride-8 model. Synchronized Batch Normalization [66] 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 [66, 69, 75].

We first consider the Cityscapes [11] 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 coarse labeled images or any extra data in this benchmark. Our ResNeSt backbone boosts the mIoU achieved by DeepLabV3 models by around 1% while

	Method	Backbone	pixAcc%	mIoU%
Prior Work	UperNet [59]	ResNet101	81.01	42.66
	PSPNet [69]	ResNet101	81.39	43.29
	EncNet [66]	ResNet101	81.69	44.65
	CFNet [67]	ResNet101	81.57	44.89
	OCNet [64]	ResNet101	-	45.45
	ACNet [16]	ResNet101	81.96	45.90
Ours	DeeplabV3 [9]	ResNet50 [19]	80.39	42.1
		ResNet101 [19]	81.11	44.14
		ResNeSt-50 (ours)	81.17	45.12
		ResNeSt-101 (ours)	**82.07**	**46.91**
		ResNeSt-200 (ours)	82.45	48.36

Table 7. Semantic segmentation results on validation set of: ADE20K.

	Method	Backbone	mIoU%
Prior Work	DANet [15]	ResNet101	77.6
	PSANet [70]	ResNet101	77.9
	PSPNet [69]	ResNet101	78.4
	AAF [31]	ResNet101	79.2
	DeeplabV3 [9]	ResNet101	79.3
	OCNet [64]	ResNet101	80.1
Ours	DeeplabV3 [9]	ResNet50 [19]	78.72
		ResNet101 [19]	79.42
		ResNeSt-50 (ours)	79.87
		ResNeSt-101 (ours)	**80.42**
		ResNeSt-200 (ours)	82.7

Table 8. Semantic segmentation results on validation set of Citscapes. Models are trained without coarse labels or extra data.

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 [73] 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 8 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.

8. Conclusion

This work proposes the ResNeSt architecture that leverages the channel-wise attention with multi-path representation into a single unified Split-Attention block, which universally improves the learned feature representations to boost performance across multiple computer vision tasks.

References

[1] Tensorflow Efficientnet. https://github.com/tensorflow/tpu/tree/master/models/official/efficientnet. Accessed: 2020-03-04. 6

[2] Sean Bell, C Lawrence Zitnick, Kavita Bala, and Ross Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2874–2883, 2016. 1

[3] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019. 2

[4] 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*, pages 6154–6162, 2018. 7, 8

[5] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: High quality object detection and instance segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019. 8

[6] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, et al. Hybrid task cascade for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4974–4983, 2019. 8

[7] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 7, 8

[8] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv:1606.00915*, 2016. 8

[9] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017. 8

[10] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*, 2015. 6

[11] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016. 8

[12] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 113–123, 2019. 5

[13] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017. 7

[14] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009. 4

[15] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual Attention Network for Scene Segmentation. 2019. 8

[16] Jun Fu, Jing Liu, Yuhang Wang, Yong Li, Yongjun Bao, Jinhui Tang, and Hanqing Lu. Adaptive context network for scene parsing. In *Proceedings of the IEEE international conference on computer vision*, pages 6748–6757, 2019. 8

[17] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Dropblock: A regularization method for convolutional networks. In *Advances in Neural Information Processing Systems*, pages 10727–10737, 2018. 6

[18] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017. 5

[19] Jian Guo, He He, Tong He, Leonard Lausen, Mu Li, Haibin Lin, Xingjian Shi, Chenguang Wang, Junyuan Xie, Sheng Zha, et al. Gluoncv and gluonnlp: Deep learning in computer vision and natural language processing. *Journal of Machine Learning Research*, 21(23):1–7, 2020. 6, 8

[20] Jian Guo, He He, Tong He, Leonard Lausen, Mu Li, Haibin Lin, Xingjian Shi, Chenguang Wang, Junyuan Xie, Sheng Zha, Aston Zhang, Hang Zhang, Zhi Zhang, Zhongyue Zhang, Shuai Zheng, and Yi Zhu. Gluoncv and gluonnlp: Deep learning in computer vision and natural language processing. *Journal of Machine Learning Research*, 21(23):1–7, 2020. 7

[21] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. *arXiv preprint arXiv:1703.06870*, 2017. 7, 8

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015. 2, 3, 4, 6

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. 4, 6

[24] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks to train convolutional neural networks for image classification. *arXiv preprint arXiv:1812.01187*, 2018. 4

[25] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 558–567, 2019. 4, 5, 6, 7

[26] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. *arXiv preprint arXiv:1709.01507*, 2017. 1, 2, 3, 4, 6, 7

[27] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018. 2, 3, 4

[28] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993*, 2016. 4, 5

[29] David H Hubel and Torsten N Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106, 1962. 1

[30] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015. 6

[31] Tsung-Wei Ke, Jyh-Jing Hwang, Ziwei Liu, and Stella X. Yu. Adaptive Affinity Fields for Semantic Segmentation. In *European Conference on Computer Vision (ECCV)*, 2018. 8

[32] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 2

[33] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 2

[34] Mu Li. *Scaling distributed machine learning with system and algorithm co-design*. PhD thesis, PhD thesis, Intel, 2017. 5

[35] Xiang Li, Wenhai Wang, Xiaolin Hu, and Jian Yang. Selective kernel networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 510–519, 2019. 1, 2, 3, 4

[36] Haibin Lin, Hang Zhang, Yifei Ma, Tong He, Zhi Zhang, Sheng Zha, and Mu Li. Dynamic mini-batch sgd for elastic distributed training: Learning in the limbo of resources. *arXiv preprint arXiv:1904.12043*, 2019. 5

[37] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013. 2

[38] 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*, pages 2117–2125, 2017. 7

[39] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 6

[40] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018. 2

[41] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010. 6

[42] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European conference on computer vision*, pages 483–499. Springer, 2016. 1

[43] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*, 2018. 2

[44] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10428–10436, 2020. 2, 4

[45] Rishi Rajalingham and James J DiCarlo. Reversible inactivation of different millimeter-scale regions of primate it results in different patterns of core object recognition deficits. *Neuron*, 102(2):493–505, 2019. 1

[46] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789, 2019. 7

[47] 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*, pages 91–99, 2015. 7

[48] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2, 4

[49] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014. 6

[50] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015. 2

[51] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017. 6

[52] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 1, 2, 6

[53] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016. 1, 5, 6

[54] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019. 2, 4, 6, 7

[55] Alvin Wan, Xiaoliang Dai, Peizhao Zhang, Zijian He, Yuandong Tian, Saining Xie, Bichen Wu, Matthew Yu, Tao Xu, Kan Chen, et al. Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12965–12974, 2020. 4

[56] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1

[57] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10734–10742, 2019. 2

[58] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019. 7, 8

[59] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. *arXiv preprint arXiv:1807.10221*, 2018. 8

[60] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *arXiv preprint arXiv:1611.05431*, 2016. 1, 3, 4, 6, 7

[61] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017. 2

[62] Zhicheng Yan, Xiaoliang Dai, Peizhao Zhang, Yuandong Tian, Bichen Wu, and Matt Feiszli. Fp-nas: Fast probabilistic neural architecture search. *arXiv preprint arXiv:2011.10949*, 2020. 4

[63] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015. 8

[64] Yuhui Yuan, Xilin Chen, and Jingdong Wang. Object-contextual representations for semantic segmentation. *arXiv preprint arXiv:1909.11065*, 2019. 8

[65] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 5

[66] Hang Zhang, Kristin Dana, Jianping Shi, Zhongyue Zhang, Xiaogang Wang, Ambrish Tyagi, and Amit Agrawal. Context encoding for semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 7, 8

[67] Hang Zhang, Han Zhang, Chenguang Wang, and Junyuan Xie. Co-occurrent features in semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 548–557, 2019. 8

[68] Xingcheng Zhang, Zhizhong Li, Chen Change Loy, and Dahua Lin. Polynet: A pursuit of structural diversity in very deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 718–726, 2017. 6

[69] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 8

[70] Hengshuang Zhao, Yi Zhang, Shu Liu, Jianping Shi, Chen Change Loy, Dahua Lin, and Jiaya Jia. PSANet: Point-wise Spatial Attention Network for Scene Parsing. In *European Conference on Computer Vision (ECCV)*, 2018. 8

[71] Li Zhaoping and Zhaoping Li. *Understanding vision: theory, models, and data*. Oxford University Press, USA, 2014. 1

[72] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *Advances in Neural Information Processing Systems*, pages 487–495, 2014. 1

[73] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proc. CVPR*, 2017. 8

[74] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9308–9316, 2019. 8

[75] Yi Zhu, Karan Sapra, Fitsum A. Reda, Kevin J. Shih, Shawn Newsam, Andrew Tao, and Bryan Catanzaro. Improving Semantic Segmentation via Video Propagation and Label Relaxation. 2019. 8

[76] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018. 7