Efficient Scale-Permuted Backbone with Learned Resource Distribution

Xianzhi Du, Tsung-Yi Lin, Pengchong Jin, Yin Cui
Mingxing Tan, Quoc Le, and Xiaodan Song

Google Research, Brain Team
{xianzhi,tsungyi,pengchong,yincui,tanmingxing,qvl,xiaodansong}@google.com

Abstract. Recently, SpineNet has demonstrated promising results on object detection and image classification over ResNet model. However, it is unclear if the improvement adds up when combining scale-permuted backbone with advanced efficient operations and compound scaling. Furthermore, SpineNet is built with a uniform resource distribution over operations. While this strategy seems to be prevalent for scale-decreased models, it may not be an optimal design for scale-permuted models. In this work, we propose a simple technique to combine efficient operations and compound scaling with a previously learned scale-permuted architecture. We demonstrate the efficiency of scale-permuted model can be further improved by learning a resource distribution over the entire network. The resulting efficient scale-permuted models outperform state-of-the-art EfficientNet-based models on object detection and achieve competitive performance on image classification and semantic segmentation.

Keywords: Scale-Permuted Model, Object Detection

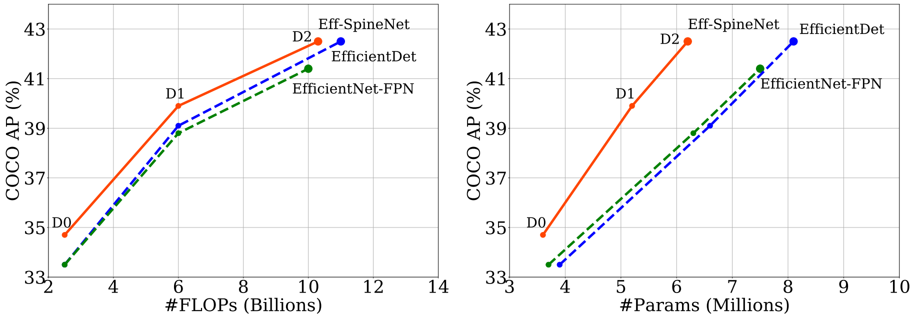

Fig. 1: Eff-SpineNet achieves better FLOPs *vs*. AP and Params *vs*. AP tradeoff curves for regular-size object detection comparing to state-of-the-art scale-decreased models EfficientNet-FPN and EfficientDet. All models adopt the RetinaNet framework [15]

1 Introduction

The scale-permuted network proposed by Du *et al.* [4] opens up the design of a new family of meta-architecture that allows wiring features with a scale-permuted ordering in convolutional neural network. The scale-permuted architecture achieves promising results on visual recognition and localization by significantly outperforming its scale-decreased counterpart when using the same residual operations but different architecture topology. Concurrently, EfficientNet-based models [23,24] demonstrate state-of-the-art performance using an advanced MBconv operation and the compound model scaling rule, while still adopting a scale-decreased backbone architecture design. A natural question is: *can we obtain new state-of-the-art performance by combining scale-permuted architecture and efficient operations?*

In this paper, we decompose the model design into three parts: (1) architecture topology; (2) operation; (3) resource distribution. The architecture topology describes the wiring and the resolution of features. The operation defines the transformation (*e.g.*, convolution and ReLU) applied to the features. The resource distribution indicates the computation allocated for each operation. Our study begins with directly combining the scale-permuted architecture topology from [4] and efficient operations from [23]. Unlike the previous works, we purposely *do not* perform any neural architecture search because the architecture topology and operation have been extensively studied and learned by sophisticated neural architecture search algorithms respectively. Instead of designing a joint search space for learning an even more tailored model, we are curious if the scale-permuted architecture and efficient operations are *generic* in the status quo and can directly be used to build the state-of-the-art model.

Despite having the learned advanced architecture topology and operation, the resource distribution has not been well studied in isolation in existing works. In [4], the resource distribution is nearly uniform for all operations, regardless of the resolution and location of a feature in the architecture. In [23], the search space contains only a few hand-selected feature dimensions for each operation and the neural architecture search algorithm is learned to select the best one. This greatly limits the possible resource distribution over the entire network. In this work, we propose a search algorithm that learns the resource distribution with the fixed architecture topology and operation. Given the target resource budget, we propose to learn the percentage of total computation allocated to each operation. In contrast to learning the absolute feature dimension, our resource targeted algorithm has the advantage of exploring a wider range of resource distribution in a manageable search space size.

We mainly conduct experiments on object detection using COCO dataset [16]. We carefully study the improvements brought by the architecture topology and operation and discover that simply combining scale-permuted architecture and MBConv operation outperforms EfficientDet [24]. The experiment results show that the architecture topology and operation are complementary for improving performance. We show that the scale-permuted EfficientNet backbone, which shares the same operation but different architecture topology with EfficientNet-

FPN, improves the performance across various models and input image sizes while using less parameters and FLOPs. We further improve the performance by learning a resource distribution for the scale-permuted EfficientNet backbone. The final model is named Efficient SpineNet (Eff-SpineNet). We discover that the model prefers to distribute resources unevenly to each operation. Surprisingly, the best resource distribution saves 18% of model parameters given the similar FLOPs, allowing us to build a more compact model.

Lastly, we take Eff-SpineNet and evaluate its performance on image classification and semantic segmentation. Eff-SpineNet achieves competitive results on both tasks. Interestingly, we find that Eff-SpineNet is able to retain the performance with less parameters. Compared with EfficientNet that is specifically designed for image classification, Eff-SpineNet has around 35% less parameters under the same FLOPs, while the Top-1 ImageNet accuracy drops by less than 1-2%. For semantic segmentation, Eff-SpineNet models achieve comparable mIOU on PASCAL VOC val 2012 to popular semantic segmentation networks, such as the DeepLab family [1,2], while using 95% less FLOPs. To summarize, these observations show that Eff-SpineNet is versatile and is able to transfer to other visual tasks including image classification and semantic segmentation.

2 Related Work

Scale-permuted network: Multi-scale feature representations have been the core research topic for object detection and segmentation. The dominant paradigm is to have a strong backbone model with a lightweight decoder such as feature pyramid networks [14]. Recently, many work has discovered performance improved with a stronger decoder [6,18,26]. Inspired by NAS-FPN [6], SpineNet [4] proposes a scale-permuted backbone architecture that removes the distinction of encoder and decoder and allows the scales of intermediate feature maps to increase or decrease anytime, and demonstrates promising performance on object detection and image classification. Auto-DeepLab [17] is another example that builds scale-permuted models for semantic segmentation.

Efficient operation: The efficiency is the utmost important problem for mobile-size convolution model. The efficient operations have been extensively studied in the MobileNet family [21,10,10,22]. Spare depthwise convolution and the inverted bottleneck block are the core ideas for efficient mobile size network. MnasNet [22] and EfficientNet [23] takes a step further to develop MBConv operation based on the mobile inverted bottleneck in [21]. EfficientNet shows that the models with MBConv operations not only achieving the state-of-the-art in ImageNet challenge but also very efficient. Recently, EfficientDet [24] builds object detection models based on the EfficientNet backbone model and achieves impressive detection accuracy and computation efficiency.

Resource-aware neural architecture search: In neural architecture search, adding resource constraints is critical to avoid the bias to choose a model with

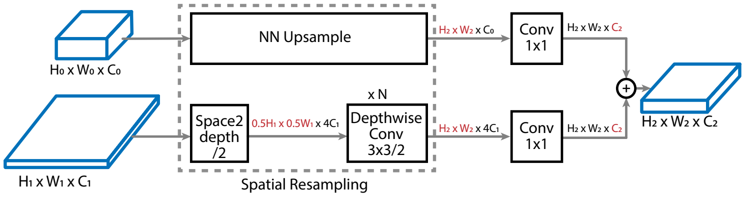

Fig. 2: Resampling operation

higher computation. MnasNet [22] introduces multi-objective rewards that optimize the model accuracy while penalizes models that violate the constraints. CR-NAS [13] searches for the best resource allocation by learning the number of blocks allocated in each resolution stage and the dilated convolution kernel.

3 Method

In this section, we first describe how to combine the scale-permuted architecture topology [4] and efficient operation MBConv [23]. Then, we introduce feature resampling and fusion operations in the efficient scale-permuted model. Lastly, we propose a search method to learn resource distribution for building Eff-SpineNet.

3.1 Scale-permuted Architecture with Efficient Operations

We first combine SpinetNet-49 architecture topology with MBconv blocks. We start with permuting the EfficientNet-B0 model. The goal here is to build an efficient scale-permuted model, SP-EfficientNet-B0, that has the similar computation and parameters as the EfficientNet-B0 baseline. We follow the idea of the compound scaling rule in EfficientNet to create 5 higher capacity models.

SP-EfficientNet-B0: We attempt to replace all the residual and bottleneck blocks in SpineNet-49 with MBconv blocks. One design decision is how to assign the convolution kernel size and feature dimension when applying MBConv to scale-permuted architecture. Given SpineNet-49 has already had a large receptive field, we decide to fix the kernel size to 3 for all MBConv operations. To obtain a model with similar size as EfficientNet-B0, we obtain the feature dimension for each level by averaging the feature dimensions over all blocks at the corresponding levels in Efficient-B0. Since the L_6 and L_7 blocks does not have a corresponding feature in EfficientNet, we follow [4] to set them to have the same feature dimension as the L_5 block. The detailed network specifications of the SP-EfficientNet-B0 model is presented in Table 1.

Compound scaling for scale-permuted network: We follow the compound scaling rule proposed in [23] to scale up the SP-EfficientNet-B0 model. We use the rule to compute the number of blocks for each feature level, feature dimension, and input image size. Since the number of blocks for a level after scaling

Table 1: Block specifications for EfficentNet-B0, SP-EfficientNet-B0, and Eff-SpineNet-D0, including block level, kernel size, and output feature dimension. SP-EfficientNet-B0 and Eff-SpineNet-D0 share same specifications for block level and kernel size

block id	EfficientNet-B0			scale-permuted models			
	level	kernel	feat. dim	level	kernel	feat. dim	
						SP-EfficientNet-B0	Eff-SpineNet-D0
1	L_1	3×3	16	L_1	3×3	16	16
2	L_2	3×3	24	L_2	3×3	24	24
3	L_2	3×3	24	L_2	3×3	24	16
4	L_3	5×5	40	L_2	3×3	24	16
5	L_3	5×5	40	L_4	3×3	96	104
6	L_4	3×3	80	L_3	3×3	40	48
7	L_4	3×3	80	L_4	3×3	96	120
8	L_4	3×3	80	L_6	3×3	152	40
9	L_4	5×5	112	L_4	3×3	96	120
10	L_4	5×5	112	L_5	3×3	152	168
11	L_4	5×5	112	L_7	3×3	152	96
12	L_5	5×5	192	L_5	3×3	152	192
13	L_5	5×5	192	L_5	3×3	152	136
14	L_5	5×5	192	L_4	3×3	96	104
15	L_5	5×5	192	L_3	3×3	40	40
16	L_5	3×3	320	L_5	3×3	152	136
17	-	-	-	L_7	3×3	152	136
18	-	-	-	L_6	3×3	152	40

may be more than the blocks at the corresponding level in SP-EfficientNet-B0 model, we uniformly repeat the blocks in SP-EfficientNet-B0 model. If the scaled number of blocks is not the multiple of those in SP-EfficientNet-B0, we add the remainder blocks one-by-one in the bottom up ordering. The detailed model scaling specifications are given in Table 2.

3.2 Feature Resampling and Fusion

Given the MBConv output feature dimension is much lower compared to residual and bottleneck blocks, we redesign the feature resampling method. And we adopt the fusion method from EfficientDet [24].

Resampling method: Since MBConv has a small output feature dimension, it removes the need of the scaling factor α in SpineNet to reduce the computation. Compared to SpineNet, the 1x1 convolution to reduce input feature dimension is removed. Besides, we find using the space-to-depth operation followed by stride 2 convolutions preserves more information than the original design, with a small increase of computation. The new resampling strategy is shown in Figure 2.

Table 2: Model scaling method for Eff-SpineNet models. **input size:** Input resolution. **feat. mult.:** Feature dimension multiplier for convolutional layers in backbone. **block repeat:** Number of repeats for each block in backbone. The 18 blocks are ordered from left to right. **feat. dim.:** Feature dimension for separable convolutional layers in subnets. **#layers:** Number of separable convolutional layer in subnets

model id	scale-permuted backbone			subnets	
	input size	feat. mult.	block repeat	feat. dim.	#layers
M0	256	0.4	{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1}	24	3
M1	384	0.5	{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1}	40	3
M2	384	0.8	{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1}	64	3
D0	512	1.0	{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1}	64	3
D1	640	1.0	{2,2,1,1,2,2,2,1,1,2,1,2,1,1,1,1,1,1}	88	3
D2	768	1.1	{2,2,1,1,2,2,2,1,1,2,1,2,1,1,1,1,1,1}	112	3

Weighted block fusion: As shown in [24], input features at different resolutions or network building stages may contribute unequally during feature fusion. We apply the fast normalized fusion strategy introduced in [24] to block fusion in SpineNet. The method is shown in Equation 1:

$$B^{out} = \frac{\sum_i w_i \times B_i^{in}}{\sum_j w_j + 0.001}, \tag{1}$$

where B^{in} and B^{out} represent the input blocks and the output block respectively. w is the weight to be learned for each input block.

3.3 Learning Resource Distribution

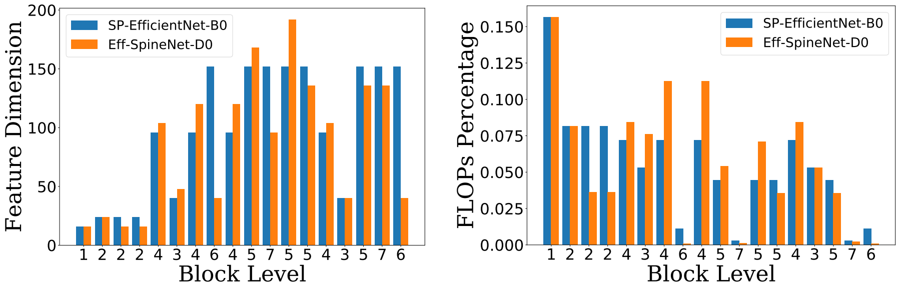

Fig. 3: Comparisons of SP-EfficientNet-B0 and Eff-SpineNet-D0 in feature dimension distribution (left) and resource distribution (right). The 18 blocks are plotted in order from left to right with block level shown in the x-axis

Typically, the conventional architecture gradually increases feature dimension with decreasing spatial resolution of feature [9,25,23,21,10]. However, this design may be sub-optimal for a scale-permuted network. In this section, we propose a simple yet effective search method for resource reallocation. We learn the resource distribution through adjusting the feature dimension of MBConv blocks. In the search space, we fix the total FLOPs of MBConv blocks in the entire model, and learn a scale multiplier of feature dimension for each block in SP-EfficientNet-B0.

Consider c_i to be the feature dimension of MBConv block i, the FLOPs can be computed as $\mathcal{F}_i \simeq C_i \times c_i^2$, where C_i is a constant that depends on height, width, and expansion ratio of a given block.

$$\begin{aligned} \mathcal{F}_i &= H_i \times W_i \times (2 \times c_i^2 \times r + k^2 \times c_i \times r) \\ &\simeq H_i \times W_i \times c_i^2 \times 2 \times r \\ &\simeq C_i \times c_i^2, \end{aligned} \tag{2}$$

where H_i, W_i is the height and width of the feature map, r is the expansion ratio and k is the kernel size in a MBConv block.

We propose to learn a multiplier α_i that adjusts the resource distribution over the entire model with a target total desired computation $\mathcal{F}_t$

$$\mathcal{F}_t = \sum_i \alpha_i \mathcal{F}_i \tag{3}$$

In our experiment, we simply set $\mathcal{F}_t = \sum_i \mathcal{F}_i$.

Learning α_i can be challenging because α_i can be any positive real number. Here, we propose to learn a multiplier β_i which is selected from a set of N positive numbers $\{\beta^1, \beta^2, ..., \beta^N\}$. Then, we can represent α_i as a function of β_i which satisfies the equation 3.

$$\alpha_i = \frac{\mathcal{F}_t}{\sum_k \beta_k \mathcal{F}_k} \beta_i \tag{4}$$

Finally, we use α_i to modify the feature dimension for each block $\hat{c}_i = \sqrt{\alpha_i} c_i$.

Using this resource distribution learning strategy, we discover our final model, Eff-SpineNet-D0. We show the model specification in Table 1 and the comparison with SP-EfficientNet-B0 in Figure 3.

4 Applications

4.1 Object Detection

We use Eff-SpineNet as backbone in RetinaNet [15] for one-stage object detection and in Mask R-CNN [8] for two-stage object detection and instance segmentation. The feature map of the 5 output blocks $\{P_3, P_4, P_5, P_6, P_7\}$ are used as the multi-scale feature pyramid. Similar to [24], we design a heuristic scaling rule to maintain a balance in computation between backbone and subnets during model scaling and use separable convolutions in all subnets. In RetinaNet, we gradually use more convolutional layers and a larger feature dimension for each layer in the box and class subnets for a larger Eff-SpineNet model. In Mask R-CNN, the same scaling rule is applied to convolutional layers in the RPN, Fast R-CNN and mask subnets. In addition, a fully connected layer is added after convolutional layers in the Fast R-CNN subnet and we apply the scaling to adjust its dimension to 256 for D0 and D1, and 512 for D2. Details are shown in Table 2.

4.2 Image Classification

We directly utilize all feature maps from P_3 to P_7 to build the classification model. Different from the object detection models shown in Table 2, we set the feature dimension to 256 for all models. The final feature vector is generated by nearest-neighbor upsampling and averaging all feature maps to the same size as P_3 followed by the global average pooling. We apply a linear classifier on the 256-dimensional feature vector and train the classification model with softmax cross-entropy loss.

4.3 Semantic Image Segmentation

In this subsection, we explore Eff-SpineNet for the task of semantic image segmentation. We apply nearest-neighbor upsampling to match the sizes of all feature maps in $\{P_3, P_4, P_5, P_6, P_7\}$ to $P3$ then take the average. The averaged feature map P at output stride 8 is used as the final feature map from Eff-SpineNet. We further apply separable convolutional layers before the pixel-level prediction layer. The number of layers and feature dimension for each layer are fixed to be the same as the subnets in object detection, shown in Table 2.

5 Experimental Results

We present experimental results on object detection, image classification, and semantic segmentation to demonstrate the effectiveness and generality of the proposed Eff-SpineNet models. For object detection, we evaluate Eff-SpineNet on COCO bounding box detection [16]. We train all models on the COCO `train2017` split and report results on the COCO `val2017` split. For image classification, we train Eff-SpineNet models on ImageNet ILSVRC-2012 and report Top-1 and Top-5 validation accuracy. For semantic segmentation, we follow the common practice to train Eff-SpineNet on PASCAL VOC 2012 with augmented 10,582 training images and report mIOU on 1,449 `val` set images.

5.1 Object Detection

5.1.1 Experimental Settings

Table 3: **One-stage object detection results on the COCO benchmark.** We compare using different backbones with RetinaNet on single model without test-time augmentation. FLOPs is represented by Multi-Adds

model	#FLOPs	#Params	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Eff-SpineNet-D0	**2.5B**	**3.6M**	**34.7**	**53.1**	**37.0**	**15.2**	**38.7**	**52.8**
EfficientNet-B0-FPN	2.5B	3.7M	33.5	52.8	35.4	14.5	37.5	50.7
EfficientDet-D0 [24]	2.5B	3.9M	33.5	-	-	-	-	-
Eff-SpineNet-D1	**6.0B**	**5.2M**	**39.9**	**59.6**	**42.5**	**43.5**	**21.1**	**57.5**
EfficientNet-B1-FPN	5.8B	6.3M	38.8	59.1	41.4	20.2	43.0	55.7
EfficientDet-D1 [24]	6.0B	6.6M	39.1	-	-	-	-	-
Eff-SpineNet-D2	**10.3B**	**6.2M**	**42.5**	**62.0**	**46.0**	**24.5**	**46.4**	**57.6**
EfficientNet-B2-FPN	10.0B	7.5M	41.4	62.3	44.1	24.4	45.4	56.8
EfficientDet-D2 [24]	11.0B	8.1M	42.5	-	-	-	-	-
ResNet-50-FPN [4]	96.8B	34.0M	42.3	61.9	45.9	23.9	46.1	58.5
SpineNet-49 [4]	85.4B	28.5M	44.3	63.8	47.6	25.9	47.7	61.1

Table 4: Ablation studies on advanced training strategies for Eff-SpineNet-D2 and ResNet50-FPN. We begins with 72 epochs training steps and multi-scale training [0.8, 1.2] as the baseline. **SE:** squeeze and excitation; **ms train:** large-scale multi-scale [0.1, 2.0] and extened training steps that attain the best performance (650 epochs for Eff-SpineNet-D2 and 250 epochs for ResNet-50-FPN); **Swish:** Swish activation that replaces ReLU; **SD:** stochastic depth

model	baseline	+SE	+ms train	+Swish	+SD
Eff-SpineNet-D2	32.2	32.6(+0.4)	40.1(+7.4)	42.1 (+2.0)	42.5(+0.4)
ResNet-50-FPN	37.0	N/A	40.4 (+3.4)	40.7 (+0.3)	42.3(+1.6)

Training details: We generally follow the training protocol in [4,24] to train all models for the proposed method, EfficientNet-FPN, and SpineNet on COCO `train2017` from scratch. We train on Cloud TPU v3 devices using standard stochastic gradient descent (SGD) with 4e-5 weight decay and 0.9 momentum. We apply batch size 256 and stepwise learning rate with 0.28 initial learning rate that decays to 0.1× and 0.01× at the last 30 and 10 epochs. All models are trained for 650 epochs, which we observe the model starts to overfit and hurt performance after 650 epochs. We apply synchronized batch normalization with 0.99 momentum, swish activation [19], and stochastic depth [12]. To preprocess training data, we resize the long side of an image to the target image size described in Table 2 then pad the short side with zeros to make it a square image. Horizontal flipping and multiscale augmentation [0.1, 2.0] are implemented during training.

Search details: We design our search space $\{\beta_1, \beta_2, ..., \beta_N\}$ as $\{1, 5, 10, 15, 20\}$ in this work to cover a wide range of possible resource distributions with a

Table 5: Impact of longer training schedule using advanced training strategies when training a model from scratch

model	72 epoch	200 epoch	350 epoch	500 epoch	650 epoch
Eff-SpineNet-D2	34.8	40.0 (+5.2)	41.4 (+1.4)	42.1 (+0.7)	42.5 (+0.4)

Table 6: **Two-stage object detection results on COCO.** We compare using different backbones with Mask R-CNN on single model

model	#FLOPs	#Params	AP	AP_{50}	AP_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}
Eff-SpineNet-D0	4.7B	4.6M	35.0	54.0	37.3	30.5	50.2	32.2
Eff-SpineNet-D1	9.2B	6.4M	40.7	60.9	44.1	35.0	56.9	36.8
Eff-SpineNet-D2	16.0B	9.2M	42.9	63.5	46.5	37.3	60.2	39.1

manageable search space size. We follow [23,4] to implement the reinforcement learning based search method [27]. In brief, we reserve 7392 images from COCO `train2017` as the validation set for searching and use other images for training. Sampled models at the D0 scale are used for proxy task training with the same training settings described above. AP on the reserved set of proxy tasks trained for 4.5k iterations is collected as rewards. The best architecture is collected after 5k architectures have been sampled.

5.1.2 Object Detection Results

RetinaNet: Our main results are presented on the COCO bounding box detection task with RetinaNet. Compared to our architecture-wise baseline EfficientNet-FPN models, our models consistently achieve 1-2% AP gain from scale D0 to D2 while using less computations. The FLOPs *vs*. AP curve and the Params *vs*. AP curve among Eff-SpineNet and other state-of-the-art one-stage object detectors are shown in Figure 1 and Figure 4 respectively.

Mask R-CNN: We evaluate Eff-SpineNet models with Mask R-CNN on the COCO bounding box detection and instance-level segmentation task. The results of Eff-SpineNet D0, D1, and D2 models are shown in Table 6.

5.1.3 Mobile-size Object Detection Results

The results of Eff-SpineNet-M0/1/2 models are presented in Table 7 and the FLOPs *vs*. AP curve is plotted in Figure 1. Eff-SpineNet models are able to consistent use less resources while surpassing all other state-of-the-art mobile-size object detectors by large margin. In particular, our Eff-SpineNet-M2 achieves 29.2% AP with 0.97B FLOPs, attaining the new state-of-the-art for mobile-size object detection.

Table 7: **Mobile-size object detection results on COCO.** Eff-SpineNet models achieve the new state-of-the-art FLOPs *vs.* AP trade-off curve

backbone model	#FLOPs	#Params	AP	AP_S	AP_M	AP_L
Eff-SpineNet-M0	**0.15B**	**0.67M**	**17.3**	**2.2**	**16.4**	**33.0**
MobileNetV3-Small-SSDLite [10]	0.16B	1.77M	16.0	-	-	-
Eff-SpineNet-M1	**0.51B**	**0.99M**	**25.0**	**7.4**	**27.3**	42.0
MobileNetV3-SSD [10]	0.51B	3.22M	22.0	-	-	-
MobileNetV2 + MnasFPN	0.53B	1.29M	23.8	-	-	-
MnasNet-A1-SSD [22]	0.8B	4.9M	23.0	3.8	21.7	42.0
Eff-SpineNet-M2	**0.97B**	**2.36M**	**29.2**	**9.7**	**32.7**	**48.0**
MobileNetV2-NAS-FPN [6]	0.98B	2.62M	25.7	-	-	-
MobileNetV2-FPN [21]	1.01B	2.20M	24.3	-	-	-

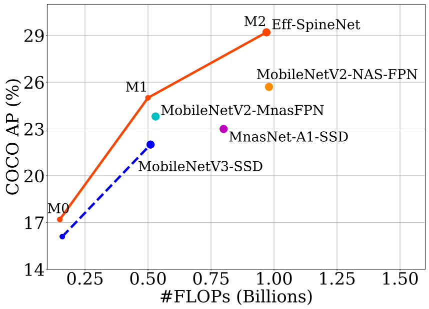

Fig. 4: A comparison of Eff-SpineNet and other state-of-the-art detectors on mobile-size object detection. Eff-SpineNet models outperform the other detectors at various scales

5.1.4 Ablation Studies

Ablation studies on advanced training strategies: We conduct detailed ablation studies on the advanced training features used in this paper and [23,4]. Starting from the final Eff-SpineNet-D2 model, we gradually remove one feature at a time: 1) removing stochastic depth in model training leads to 0.4 AP drop; 2) replacing swish activation with ReLU leads to 2.0 AP drop; 3) using less aggressive multi-scale training strategy with 72 training epochs leads to 7.5 AP drop; 4) removing squeeze and excitation [11] layers from all MBConv blocks leads to 0.4 AP drop. We further perform the ablation studies to ResNet-50-FPN and the results are shown in Table 4.

Impact of longer training schedule: We conduct experiments by adopting different training epochs for Eff-SpineNet-D2. We train all models from scratch on COCO 2017train and report AP on COCO 2017val. The results are presented in Table 5. We show that prolonging the training epochs from 72 to 650 gradu-

Table 8: Improvement from learning a better resource distribution. All models are evaluated on COCO `2017val`

model	#FLOPs	#Params	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
SP-EfficientNet-B0	2.4B	4.4M	33.0	50.3	34.7	13.0	38.4	51.7
Eff-SpineNet-D0	2.5B	3.6M	33.8	51.3	35.8	13.6	39.3	52.4

Table 9: An ablation study of the two architecture improvements in Eff-SpineNet

model	weighted fusion	space-to-depth	#FLOPs	#Params	AP
Eff-SpineNet-D0	✓	✓	2.5B	3.6M	33.8
model 1	✓	-	2.4B	3.3M	33.1
model 2	-	-	2.4B	3.3M	32.8

ally improves the performance of Eff-SpineNet-D2 by 7.7% AP. Except training schedule, the other training strategies are the same as Section 5.1.1.

Learning Resource Distribution: From the final architecture discovered by NAS shown in Table 1, we observe that parameters in low-level L_2 blocks and high-level $\{L_6, L_7\}$ block, are reallocated to middle-level $\{L_3, L_4, L_5\}$ blocks. Since the high-level blocks are low in resolution, by doing so, the number of parameters in the network is significantly reduced from 4.4M to 3.6M while the total FLOPs remains roughly the same. Learning resource distribution also brings a 0.8% AP gain. The performance improvements from SP-EfficientNet-B0 to Eff-SpineNet-D0 is shown in Table 8.

Architecture Improvements: We conduct ablation studies for the two techniques, resampling method based on the space-to-depth operation and weighted block fusion, introduced to SpineNet's scale-permuted architecture with Eff-SpineNet-D0. As shown in Table 9, the performance drops by 0.7% AP if we remove the new resampling method. The performance further drops by 0.3% AP if we remove weighted block fusion.

5.1.5 A study of the proposed search algorithm

We visualize some of the randomly sampled architectures in the search phase. The FLOPs *vs.* AP plot and the Params *vs.* AP plot are presented in Figure 5. From the FLOPs *vs.* AP plot, we can observe that the FLOPs of the sampled architectures fall into a range of ±10% of our target FLOPs because of the proposed search algorithm. We can also observe from the Params *vs.* AP plot that the number of parameters in sampled architectures are reduced in most cases.

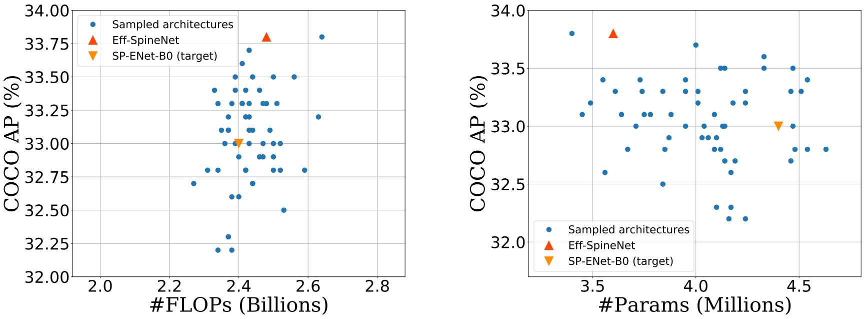

Fig. 5: FLOPs *vs*. AP (left) and Params *vs*. AP (right) plots for architectures sampled throughout the searching phase. The x-axes are plotted within a ±20% range to the centers

Table 10: Comparison between Eff-SpineNet and EfficientNet on ImageNet classification

model	input resolution	feature dim	#FLOPs	#Params	Top-1	Top-5
Eff-SpineNet-D0	224 × 224	**256**	**0.38B**	**3.57M**	**75.3**	**92.4**
EfficientNet-B0	224 × 224	1280	0.39B	5.30M	77.3	93.5
Eff-SpineNet-D1	240 × 240	**256**	**0.70B**	**4.97M**	**77.7**	**93.6**
EfficientNet-B1	240 × 240	1280	0.70B	7.80M	79.2	94.5
Eff-SpineNet-D2	256 × 256	**256**	**0.89B**	**5.83M**	**78.5**	**94.2**
EfficientNet-B2	260 × 260	1280	1.00B	9.20M	80.3	95.0

5.2 Image Classification

We conduct image classification experiments on ImageNet ILSVRC 2012 [3,20] with Eff-SpineNet, following the same training strategy used in EfficientNet [23]. We scale the input size with respect to different model size by roughly following the compound scaling [23] and adjusting it to be the closest multiples of 16.

We compare Eff-SpineNet with EfficientNet in all aspects in Table 10. At the same FLOPs, Eff-SpineNet is able to save around 35% parameters at the cost of 1.5-2% drop in top-1 accuracy. We hypothesize this is likely due to the fact that higher level features (P_6 and P_7) do not contain enough spatial resolution for small input size. For 224 × 224 input size, the spatial resolution of P_6 and P_7 is only 4 × 4 and 2 × 2 respectively. We will explore how to construct better scale-permuted models for image classification in the future.

5.3 Semantic Segmentation

We present experimental results of employing Eff-SpineNet as backbones for semantic segmentation. We conduct the experiments with evaluation metric mIOU

Table 11: Semantic segmentation result comparisons of Eff-SpineNet and other popular semantic segmentation networks on the PASCAL VOC 2012 `val` set

model	ImageNet pre-train	COCO pre-train	output stride	#FLOPs	mIOU
MobileNetv2 + DeepLabv3	-	✓	16	2.8B	75.3
ResNet-101 + DeepLabv3	✓	✓	8	81.0B	80.5
Eff-SpineNet-D0	-	✓	8	**2.1B**	**76.0**
Eff-SpineNet-D2	-	✓	8	**3.8B**	**78.0**

on PASCAL VOC 2012 [5] with extra annotated images from [7]. For training implementations, we generally follow the settings in Section 5.1.1. In brief, we fine-tune all models from the COCO bounding box detection pre-trained models for 10k iterations with batch size 256 with cosine learning rate. We set the initial learning to 0.05 and a linear learning rate warmup is applied for the first 500 iterations. We fix the input crop size to 512 × 512 for all Eff-SpineNet models without strictly following the scaling rule described in Table 2.

Our results on PASCAL VOC 2012 `val` set are presented in Table 11. Eff-SpineNet-D0 slightly outperforms MobileNetv2 with DeepLabv3 [21,1] by 0.7 mIOU while using 25% less FLOPs. Our D2 model is able to attain comparable mIOU with other popular semantic segmentation networks, such as ResNet101 with DeepLabv3 [1], at the same output stride while using 95% less FLOPs.

6 Conclusion

In this paper, we propose to decompose model design into architecture topology, operation, and resource distribution. We show that simply combining scale-permuted architecture topology and efficient operations achieves new state-of-the-art in object detection, showing the benefits of efficient operation and scale-permuted architecture are complementary. The model can be further improved by learning the resource distribution over the entire network. The resulting Eff-SpineNet is a versatile backbone model that can be also applied to image classification and semantic segmentation tasks, attaining competitive performance, proving Eff-SpineNet is a versatile backbone model that can be easily applied to many tasks without extra architecture design.

References

1. Chen, L.C., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. ArXiv **abs/1706.05587** (2017) 3, 14
2. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. Lecture Notes in Computer Science p. 833851 (2018). https://doi.org/10.1007/978-3-030-01234-2$_4$9, http://dx.doi.org/10.1007/978-3-030-01234-2_49 3
3. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR (2009) 13
4. Du, X., Lin, T.Y., Jin, P., Ghiasi, G., Tan, M., Cui, Y., Le, Q.V., Song, X.: Spinenet: Learning scale-permuted backbone for recognition and localization (2019) 2, 3, 4, 9, 10, 11
5. Everingham, M., Eslami, S.M., Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes challenge: A retrospective. Int. J. Comput. Vision **111**(1), 98136 (Jan 2015). https://doi.org/10.1007/s11263-014-0733-5, https://doi.org/10.1007/s11263-014-0733-5 14
6. Ghiasi, G., Lin, T.Y., Le, Q.V.: Nas-fpn: Learning scalable feature pyramid architecture for object detection. In: CVPR (2019) 3, 11
7. Hariharan, B., Arbelaez, P., Bourdev, L., Maji, S., Malik, J.: Semantic contours from inverse detectors. In: International Conference on Computer Vision (ICCV) (2011) 14
8. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: ICCV (2017) 7
9. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016) 7
10. 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: ICCV (2019) 3, 7, 11
11. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (Jun 2018). https://doi.org/10.1109/cvpr.2018.00745, http://dx.doi.org/10.1109/CVPR.2018.00745 11
12. Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K.Q.: Deep networks with stochastic depth. In: ECCV (2016) 9
13. Liang, F., Lin, C., Guo, R., Sun, M., Wu, W., Yan, J., Ouyang, W.: Computation reallocation for object detection. In: International Conference on Learning Representations (2020), https://openreview.net/forum?id=SkxLFaNKwB 4
14. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: CVPR (2017) 3
15. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: ICCV (2017) 1, 7
16. 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: ECCV (2014) 2, 8
17. Liu, C., Chen, L.C., Schroff, F., Adam, H., Hua, W., Yuille, A.L., Fei-Fei, L.: Autodeeplab: Hierarchical neural architecture search for semantic image segmentation. In: CVPR (2019) 3
18. Liu, S., Qi, L., Qin, H., Shi, J., Jia, J.: Path aggregation network for instance segmentation. In: CVPR (2018) 3
19. Ramachandran, P., Zoph, B., Le, Q.V.: Searching for activation functions (2017) 9

20. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. IJCV (2015) 13
21. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: CVPR (2018) 3, 7, 11, 14
22. Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., Le, Q.V.: Mnasnet: Platform-aware neural architecture search for mobile. In: CVPR (2019) 3, 4, 11
23. Tan, M., Le, Q.: EfficientNet: Rethinking model scaling for convolutional neural networks. In: Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research 2, 3, 4, 7, 10, 11, 13
24. Tan, M., Pang, R., Le, Q.V.: Efficientdet: Scalable and efficient object detection (2019) 2, 3, 5, 6, 7, 9
25. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: CVPR (2017) 7
26. Zhao, Q., Sheng, T., Wang, Y., Tang, Z., Chen, Y., Cai, L., Ling, H.: M2det: A single-shot object detector based on multi-level feature pyramid network. AAAI (2019) 3
27. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. In: ICLR (2017) 10