The supplementary materials are organized as follows. In Appendix A, we illustrate the details of experimental settings. In Appendix B, we provide the implementation details of evolutionary search. We report the effect of performance improvement with more training and searching cost in Appendix C. Then we investigate the effect of the prior initial population in Appendix D. We conduct the performance comparison of BCNet and AutoSlim [43] with the same training recipe in Appendix E. In Appendix G, we investigate the effect of training BCNet with different epochs. We report more detailed searching results of BCNet on ImageNet dataset. In Appendix I, we include more detailed searching results for Table 1. Then we present the performance of searched models w.r.t. different FLOPs in Appendix H. Finally, we show the visualization of searched network widths with $2\times$ acceleration in Appendix J.

A. Details of Training Recipe

In this section, we present the training details of our BCNet w.r.t. experiments on various models. In detail, we search on the reduced space $\mathcal{C}_K$ with default $K = 20$. During training, except for EfficientNet-B0 and ProxylessNAS, we use SGD optimizer with momentum 0.9 and nesterov acceleration. As for EfficientNet-B0 and ProxylessNAS, we adopt RMSprop optimizer for searching optimal network width.

Training Recipe of ResNet50, MobileNetV2, and VGGNet. For ResNet50, we follow the same training recipe as TAS [6]. In detail, we use a weight decay of 10^{-4} and batch size of 256; and we train the model by 120 epochs with the learning rate annealed with cosine strategy from initial value 0.1 to 10^{-5}. For MobileNetV2 and VGGNet, we set weight decay to 5×10^{-5} and 10^{-4}, respectively. Besides, for MobileNetV2, we adopt the batch size of 256, and the learning rate is annealed with a cosine strategy from initial value 0.1 to 10^{-5}. For VGGNet, we train it for 400 epochs using a batch size of 128; the learning rate is initialized to 0.1 and divided by 10 at 160-th, 240-th epoch. Moreover, we note that most pruning methods do not report their results by incorporating the knowledge distillation (KD) [17] improvement in retraining except for MobileNetV2. Thus in our method, except for MobileNetV2, we do not include KD in final retraining for a more fair comparison of performance. All experiments are implemented with PyTorch on NVIDIA V100 GPUs.

Training Recipe of EfficientNet-B0 and ProxylessNAS. We use the same training strategies for both EfficientNet-B0 and ProxylessNAS. In detail, we train both models for 300 epochs using a batch size of 1024; the learning rate is initialized to 0.128 and decayed by 0.963 for every 3 epochs. Besides, the first 5 training epochs are adopted as warm-up epochs, and the weight decay is set to 1×10^{-5}.

B. Details of Evolutionary Search

During our evolutionary search, each network width $\boldsymbol{c}$ is indicated by the averaged accuracy of its left and right paths, which can be formulated as Eq. (15). The optimal width (rather than sub-nets) refers to the one with the highest performance and we then train it from scratch.

$$\text{Accuracy}(W, \boldsymbol{c}; \mathcal{D}_{val}) = \frac{1}{2} \cdot \left(\text{Accuracy}(\mathcal{N}_l, \boldsymbol{c}; \mathcal{D}_{val}) + \text{Accuracy}(\mathcal{N}_r, \boldsymbol{c}; \mathcal{D}_{val})\right). \tag{15}$$

Concretely, we adopt the multi-objective NSGA-II [4] algorithm to implement the search. Note that some networks (*e.g.*, MobileNetV2) may have batch normalization (BN) layers, and due to the varying network widths, the mean and variance in the BN layers are not suitable to all widths. In this way, we simply use the mean and variance in batches instead, and we set the batch size to 2048 during testing to ensure accurate estimates of the mean and variance.

In detail, we set the population size as 40 and the maximum iteration as 50. Firstly, we apply our proposed prior initial population sampling method Eq. (12) $\sim$ Eq.(14) to generate the initial population. In each iteration, we use the trained BCNet to evaluate each width and rank all widths in the population. After the ranking, we use the tournament selection algorithm to select 10 widths retained in each generation. And the population for the next iteration is generated by two-point crossover and polynomial mutation. Finally, the searched width refers to the one with the best performance in the last iteration, and we train it from scratch for evaluation.

Pipeline of training and evolutionary search: We follow a routine pipeline in width searching methods, such as TAS and AutoSlim. We first train a supernet (*i.e.*, BCNet) and then use it to search for the optimal width by evolutionary algorithms. For each sampled width during the search, we evaluate it by the inference with the weights from BCNet and record its accuracy; since there is no network training during evolutionary, it is thus very efficient. Finally, we only retrain the width with the highest accuracy from scratch and report its performance.

C. Effect of Performance Improvement with more Training and Searching Cost

Since BCNet needs 2× of training and searching cost of the UA principle, one intuitive question comes to *whether UA principle can benefit from more training and searching cost and even surpass BCNet as a result.* With this aim, we search with UA principle on more times (1× and 2×) of training epochs and iterations with evolutionary search to search for 0.5× FLOPs ResNet50 and MobileNetV2 on ImageNet dataset.

Table 5. Performance with more training and searching cost on ImageNet dataset. Note BCNet achieves 76.90% and 70.20% accuracy for 0.5× FLOPs ResNet50 and MobileNetV2, respectively.

ResNet50			MobileNetV2		
SearchingTraining	1×	2×	SearchingTraining	1×	2×
1×	75.63%	75.65%	1×	68.95%	68.98%
2×	75.74%	75.72%	2×	69.03%	69.06%

From Table 5, we can know that evolutionary search only benefits a little from more searching iterations. Simultaneously, 2 times of training epochs for supernet nearly does not affect the search result. As a result, our BCNet can efficiently boost the performance of searching results with two times of training and searching cost. Note that after the search, we train the searched network width from scratch for evaluation, which amounts to the same cost as other methods [6, 43, 25].

D. Effect of Prior Initial Population Sampling (PIPS)

Our proposed prior initial population sampling (PIPS) method aims to provide a better initial population for evolutionary search, and the searched optimal width will have higher performance accordingly. Now we want to investigate how the effect of directly leveraging PIPS to search for optimal width. With this aim, we pick up the optimal width with the highest validation Top-1 accuracy after $\{100, 200, 500, 1000, 1500, 2000\}$ of search number of widths, respectively. Then we train them from scratch and report their Top-1 accuracy in Figure 6. The search is implemented on ResNet50 on ImageNet dataset with 3 different settings and 0.5× FLOPs budget, *i.e.*, evolutionary search with random initial population, evolutionary search with prior initial population, and search with the only prior initial population.

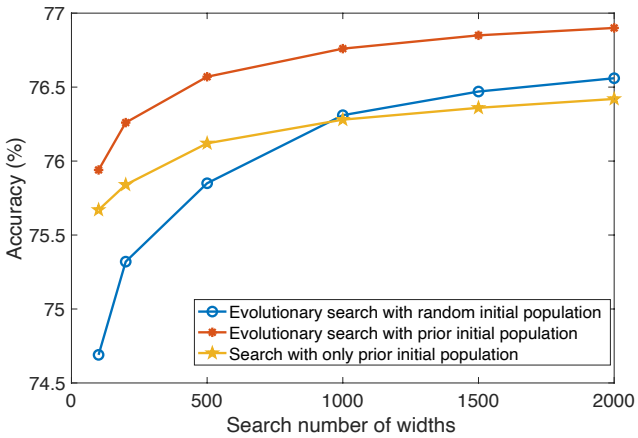


Figure 6. Top-1 accuracy of searched models on ImageNet dataset by different methods with the increasing of search numbers.

From Figure 6, we can know that widths provided by our prior initial population sampling method can surpass those from the random initial population by a larger gap on Top-1 accuracy. Besides, it also should be noticed that evolutionary search benefits more from the increase of search numbers, which indicates that evolutionary search can better utilize the searched width to achieve the optimal result. In addition, with our prior initial population, evolutionary search can get better network width with higher performance (*i.e.*, red line in Figure 8), which means that our prior initial population sampling method can provide good initialization for evolutionary algorithm. Moreover, the Top-1 accuracy of searched models rises slowly after 1000 search numbers, which may imply that the evolutionary algorithm can already find a good solution in this case.

E. Comparison of BCNet and AutoSlim [43] under 305M FLOPs

To intuitively check the effect of BCNet with another baseline method, we visualize the network width searched by BCNet and the released structure of AutoSlim [43] for 305M-FLOPs MobileNetV2 in Figure 7. In detail, BCNet saves more layer widths in the first few layers, and prunes a bit more widths in the last few layers, which is more evenly than AutoSlim.

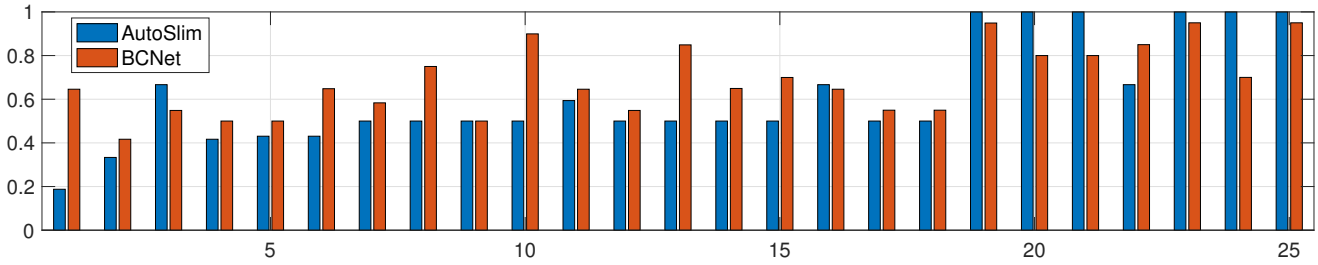


Figure 7. Visualization of searched MobileNetV2 with 305M FLOPs on ImageNet dataset. Both networks are searched on 1.5× search space as AutoSlim [43].

To promote the fair comparison of BCNet and AutoSlim, we retrain the released structure of AutoSlim (*i.e.*, 305 FLOPs MobileNetV2) with the same training recipe of ours. Note that we do not include KD for a more fair comparison of AutoSlim [43], as shown in Table 6.

Table 6. Performance comparison with AutoSlim [43] of 305M MobileNetV2 on ImageNet by the same training recipe.

Methods	FLOPs	Parameters	Top-1	Top-5
AutoSlim	305M	5.8M	73.1%	91.1%
BCNet	305M	4.8M	73.9%	92.2%

From Table 6 and Figure 7, we can know that BCNet retains more widths closer to the input layer, and thus the parameters of our searched structures are lesser than AutoSlim. Moreover, with the same training recipe, BCNet achieves a 0.8% higher on Top-1 accuracy than AutoSlim with 305M FLOPs MobileNetV2, which indicates the effectiveness of our method.

F. Transferability of the Searched Width to Object Detection Task

For object detection tasks, a pretrained model on ImageNet dataset is usually leveraged as its backbone. As a result, We take the searched 2G FLOPs ResNet50 as the backbone in detection to examine the transferability of BCNet for other tasks [23, 29]. The results are reported in Table 7 for both Faster R-CNN with FPN and RetinaNet, indicating the backbones obtained by BCNet(0.5×) can achieve higher performance to the uniform baseline(0.5×)

Table 7. Detection performance with ResNet50 as the backbone.

Framework	Original (4G)	BCNet (2G)	Uniform(2G)
RetinaNet	36.4%	35.4%	34.3%
Faster R-CNN	37.3%	36.3%	35.4%

G. Effect of Training BCNet with Different Epochs

Since BCNet is critical in searching width as a fundamental performance estimator, we investigate how much effort should be spared in its training to ensure the searching performance. In this way, we train the BCNet with a different number of epochs and use these BCNets to search MobileNetV2 with 50% FLOPs on ImageNet and CIFAR-10 datasets. The Top-1 accuracies of searched networks are presented in Figure 8.

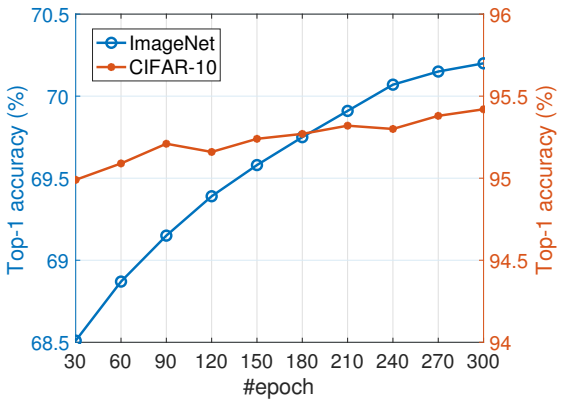
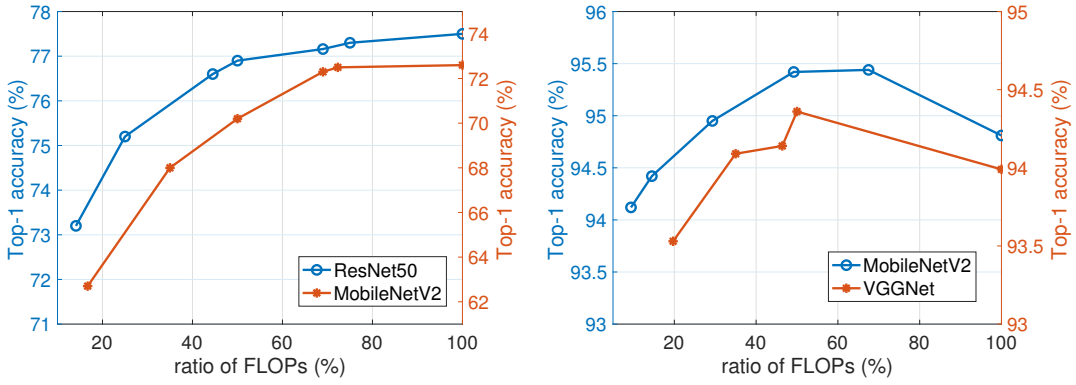


Figure 8. Performance of searched MobileNetV2 (50% FLOPs) w.r.t. different training epochs of BCNet on ImageNet and CIFAR-10 dataset.

From Figure 8, we can see that more challenging (*e.g.*, ImageNet) datasets usually need more training epochs for the performance estimator than simple datasets (*e.g.*, CIFAR-10). Concretely, with 30 (300) training epochs of BCNet, the searched MobileNetV2 can achieve 68.51% (70.20%) Top-1 accuracy on ImageNet dataset. In contrast, for CIFAR-10 dataset, the Top-1 accuracy w.r.t. 300 training epoch is merely 0.43% better than that of 30 training epochs.

H. Searching Performance with Different FLOPs

To explore the performance of searched models w.r.t. different FLOPs, we include more results on the Top-1 accuracy of ResNet50, MobileNetV2, and VGGNet searched on ImageNet and CIFAR-10 datasets, as shown in Figure 9. Note that we report the ratio of FLOPs of searched models instead of their absolute FLOPs values for clarity.



(a) ImageNet. (b) CIFAR-10.

Figure 9. Top-1 accuracy of searched models on ImageNet and CIFAR-10 dataset w.r.t. different ratios of FLOPs.

Figure 9(a) shows that the performance of searched models worsens with decreasing ratios of FLOPs on ImageNet dataset. However, the accuracy drop of ResNet50 under a small FLOPs ratio is slighter than that of MobileNetV2. For example, the Top-1 accuracy of ResNet50 (MobileNetV2) drops for 5.2% (9.9%) with 14.1% (16.7%) ratio of FLOPs. Besides, as shown in Figure 9(b), our searched MobileNetV2 and VGGNet can achieve better performance than the original model on CIFAR-

10 dataset. In details, our searched MobileNetV2 (VGGNet) with 50% ratio of FLOPs outperforms the original model (100% ratio of FLOPs) by 0.63% (0.37%) on Top-1 accuracy.

I. More Detailed Results of ImageNet for Table 1

To investigate the effect of BCNet, we further report our algorithm on ResNet34 and ResNet18 with the same training recipe as ResNet50. The original ResNet34 (ResNet18) has 21.8M (11.7M) parameters and 3.6G (1.8G) FLOPs with 74.9% (71.5%) Top-1 accuracy, respectively. As shown in Table 8, our searched 0.5× ResNet34 (ResNet18) achieve 73.3% (69.9%) FLOPs, which exceeds FPGM (DMCP) by 0.8% (0.7%) on Top-1 accuracy. Moreover, with tiny FLOPs (*i.e.*, 360M and 450M), BCNet can surpass the unform baseline by a large margin.

Table 8. Performance comparison of ResNet34, ResNet18, and MobileNetV2 on ImageNet. Methods with "*" denotes that the results are reported with knowledge distillation.

ResNet34

Groups	Methods	FLOPs	Parameters	Top-1	Top-5
2.7G	Rethinking	2.79G	-	72.9%	-
	PF	2.79G	-	72.1%	-
	MIL	2.75G	-	73.0%	-
	Uniform	2.7G	-	72.3%	90.8%
	Random	2.7G	-	71.4%	90.6%
	BCNet	2.7G	20.2M	**74.9%**	92.4%
	CNN-FCF	2.7G	15.9M	73.6%	91.5%
	BCNet	2.5G	20.0M	**74.6%**	92.2%
1.8G	FPGM	2.2G	-	72.5%	-
	SFP	2.2G	-	71.8%	90.3%
	CNN-FCF	2.2G	12.6M	72.8%	91.0%
	GS	2.1G	-	72.9%	-
	Uniform	1.8G	-	71.5%	90.2%
	Random	1.8G	-	70.9%	89.8%
	BCNet	1.8G	16.9M	**73.3%**	91.4%
	CGNet	1.8G	-	71.3%	-
	CNN-FCF	1.7G	9.6M	71.3%	90.2%
0.9G	CNN-FCF	1.2G	7.1M	69.7%	89.3%
	CGNet	1.2G	-	70.2%	-
	Uniform	0.9G	-	69.6%	89.5%
	Random	0.9G	-	68.8%	88.7%
	BCNet	0.9G	9.6M	**72.2%**	89.8%
0.36G	Uniform	0.36G	-	59.6%	82.1%
	Random	0.36G	-	56.4%	80.7%
	BCNet	0.36G	3.6M	**64.4%**	85.7%

MobileNetV2

Groups	Methods	FLOPs	Parameters	Top-1	Top-5
150M	TAS*	150M	-	70.9%	-
	LEGR	150M	-	69.4%	-
	Uniform	150M	2.0M	69.3%	88.9%
	Random	150M	-	68.8%	88.7%
	BCNet	150M	2.9M	**70.2%**	89.2%
	BCNet*	150M	2.9M	**71.2%**	89.6%

ResNet18

Groups	Methods	FLOPs	Parameters	Top-1	Top-5
1.2G	TAS*	1.2G	-	69.2%	89.2%
	MIL	1.2G	-	66.3%	86.9%
	Uniform	1.2G	8.5M	68.8%	88.5%
	Random	1.2G	-	68.4%	88.1%
	BCNet	1.2G	11.6M	**71.3%**	90.1%
1G	SFP	1.05G	-	67.1%	87.8%
	FPGM	1.04G	-	68.4%	88.5%
	DMCP	1.04G	-	69.2%	-
	CGNet	0.94G	-	68.8%	-
	DCP	0.96G	-	67.4%	87.6%
	FBS	0.9G	-	68.2%	88.2%
	Uniform	0.9G	6.0M	67.1%	87.5%
	Random	0.9G	-	66.7%	87.1%
	BCNet	0.9G	9.8M	**69.9%**	89.1%
450M	Uniform	450M	2.9M	61.6%	83.6%
	Random	450M	-	59.8%	82.3%
	BCNet	450M	4.9M	**65.8%**	86.4%

J. More Visualization of Searched Network Widths

For intuitively understanding, we visualize our searched networks as in Figure 10. Moreover, we show the retained ratio of layer widths for clarity compared to that of the original models. Note that for MobileNetV2, ResNet50, EfficientNet-B0, and ProxylessNAS with skipping or depthwise layers, we merge these layers, which are required to have the same width. From Figure 10, we notice that the width of EfficientNet-B0 and ProxylessNAS varies more evenly than other models, which means these NAS searched models have more delicate widths than others. Besides, models pruned on CIFAR-10 tend to retain a smaller width on the last layer than on ImageNet, which indicates that the difficulty of the dataset determines the width of the last layer.

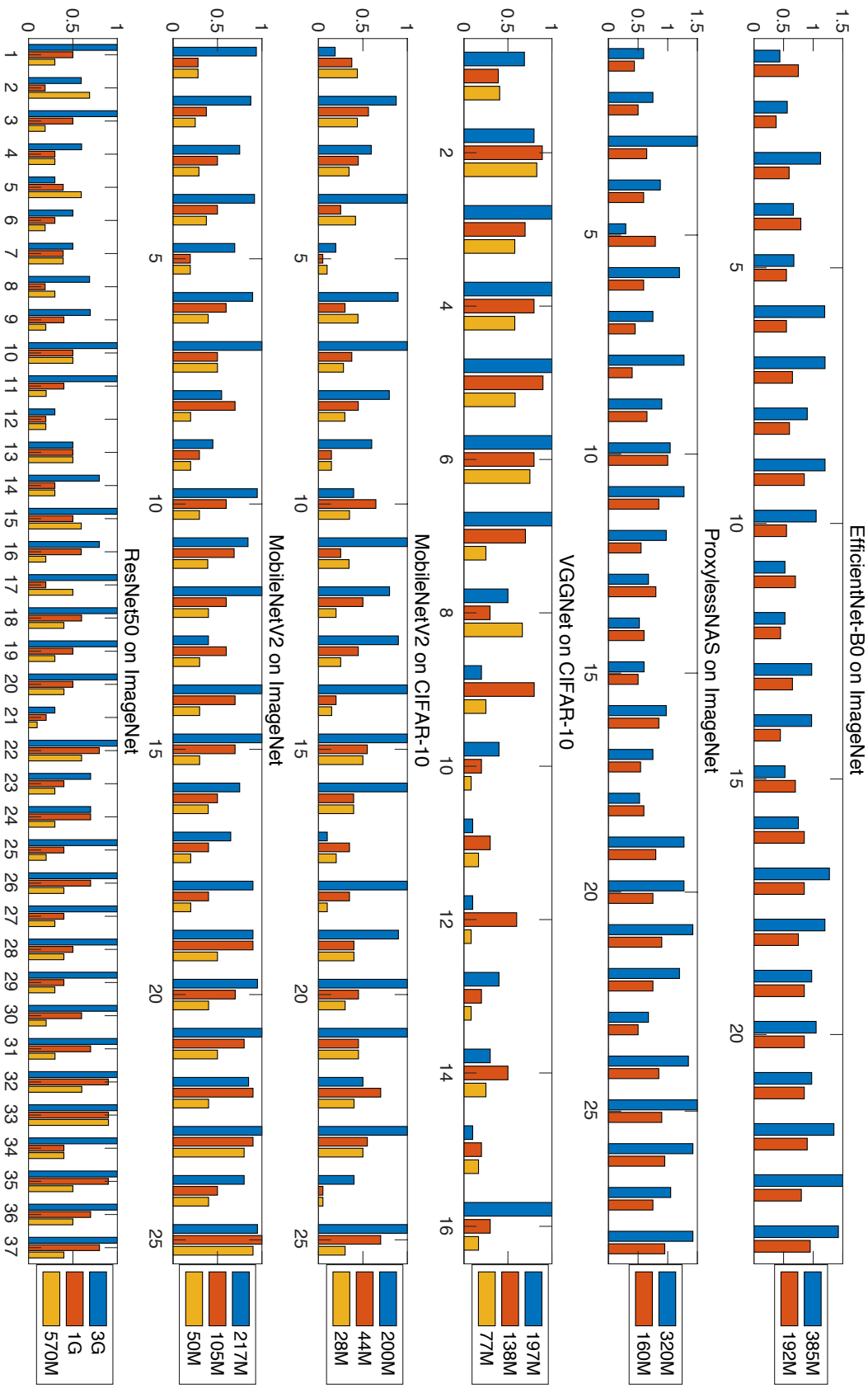


Figure 10. More visualization of searched networks w.r.t. different FLOPs. The vertical axis means the ratio of retained width compared to that of original networks at each layer.