Boosting Order-Preserving and Transferability for Neural Architecture Search: a Joint Architecture Refined Search and Fine-tuning Approach

Supplementary Material

1. Search Space

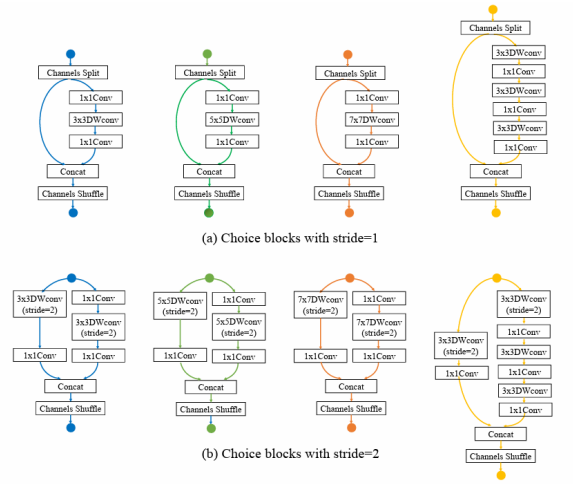

Figure 1. The choice block of our search space based on ShuffleNet-V2. From left to right: *Choice_3, Choice_5, Choice_7, Choice_x*. The figure is the same as SPOS as we use the same search space.

Table 1. Design of our supernet. CB: choice block. GAP: global average pooling. Stride: stride of the first layer in each block.

Input size	Blocks	Channels	Repeat	Stride
$224^2 \times 3$	3×3 conv	16	1	2
$112^2 \times 16$	CB	64	4	2
$56^2 \times 64$	CB	160	4	2
$28^2 \times 160$	CB	320	8	2
$14^2 \times 320$	CB	640	4	2
$7^2 \times 640$	1×1conv	1024	1	1
$7^2 \times 1024$	GAP	1024	1	-
1024	fc	1000	1	-

Fig. 1 shows our choice block and search space, which is based on *ShuffleNet-V2*, a strong lightweight convolutional neural network. Table 1 shows our supernet. The supernet helps embody our search space and provide a relative performance estimator for different architectures.

2. Searching Result

Fig. 2 shows the searched architectures on ImageNet-1K and ImageNet-100 respectively. We can see that for ImageNet-100, it gets a rather simpler architecture in the same search space comparing with ImageNet-1K. This is in line with our intuition.

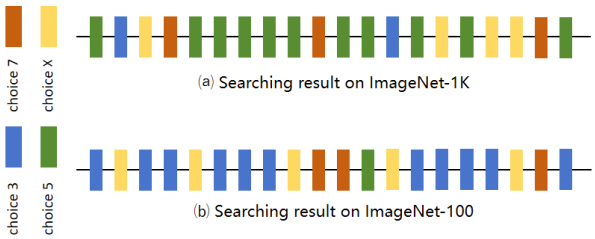

Figure 2. The searched architecture on ImageNet-1K and ImageNet-100

3. Further discussion on Order-Preserving

We find that global order-preserving ability is relatively general among different tasks while local order-preserving ability is task-specific. In other words, some architectures are in-born superior and are more likely to perform better in different tasks. Our experimental results show that the top 10 architectures searched on ImageNet-1K outperform another 10 random architectures by 0.5% on Cifar-100 and 0.9% on ImageNet-100 on average. It implies global order can be roughly preserved across datasets. However, the ranking of the top 10 architectures by the performance of Cifar-100 and ImageNet-100 is quite different from the ranking by the performance of ImageNet-1K, showing that the local relative ranking is task-specific since it can perform differently in different tasks.

This is an interesting finding and it can provide a rough estimation and a better 'start point' for searching.

4. Hyperparameter setting

For the training of supernet and the retraining of the searched architectures, we use the same setting (including hyper-parameters, data-augmentation strategy, learning-rate decay, etc.) as SPOS. The batchsize is 1024, the supernet is trained for 150,000 iterations and the searched architecture is trained for 300,000 iterations on ImageNet-1K. For ImageNet-100, the batchsize is 256, the supernet is trained for 80,000 iterations and the searched architecture is trained for 120,000 iterations.

The learning rate in supernet shifting is 1e-4. In total 640 samples are used to compute loss.

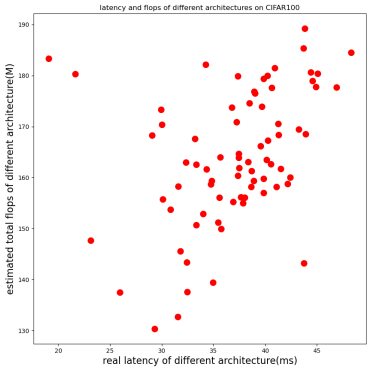

Figure 3. The correlation between flops and real-time latency on Cifar dataset.

Table 2. The retrain result of ImageNet-100 using different constraint in searching.

Constraint	Top-1 acc	Flops	Latency
Flops	85.61	299M	62.14ms
Latency	85.59	305M	58.31ms

5. Experiments on Edge Devices

Unlike cloud servers, edge devices usually have constraints on the memory and computational resources, making it intractable to load complex models. Neural architecture search is one of the most popular and effective techniques that can design efficient neural architectures for edge devices with limited resources.

Our method can be easily applied to such an on-device NAS task. Constraints on the resources, such as FLOPs and latency, can be seen as multiple objects in the evolutionary searching stage. Unlike most current NAS methods that use FLOPs to approximate the efficiency of architecture, our work also supports utilizing the latency from real-time measurement on edge devices as one of the search objects.

Specifically, we apply our method on *ROC-RK3588S-PC*, an 8-Core 8K AI Mainboard, and adopt the NSGA-II multiple object optimization method to search the Pareto superior to both accuracy and latency. The result is shown in Table 2.

From the result, we can see that those architectures which have lower Flops do not necessarily have lower latency. Therefore, we further analyze the correlation between the FLOPs and latency on a large number of architectures. Results are shown in Fig. 3. The Kendall's tau is only 0.3, which shows a non-negligible gap between Flops and real-time latency. Therefore, supporting real-time measurements is vital for applying NAS to on-device AI.