Supplementary Materials for "Breaking the Curse of Space Explosion: Towards Efficient NAS with Curriculum Search"

Yong Guo Yaofo Chen Yin Zheng Peilin Zhao Jian Chen Junzhou Huang Mingkui Tan

We organize our supplementary material as follows. First, we give more discussions on Eqn. (3) that calculates the size of search space in Section A. Second, we detail the training method of CNAS in Section B. Third, we provide more evaluation details for the inferred architectures in Section C.

A. More Discussions on Search Space Size

In this section, we focus on the cell-based architecture and analyze the effect of the number of nodes N and the number of candidate operations K on the size of search space.

In this paper, we focus on searching for the optimal cell-based architecture (Pham et al., 2018; Liu et al., 2019; Guo et al., 2019). A cell-based architecture can be represented by a directed acyclic graph (DAG), *i.e.*, $\alpha = (\mathcal{V}, \mathcal{E})$. $\mathcal{V}$ is the set of the nodes that represent the feature maps in the neural networks. $\mathcal{E}$ is the set of the edges that represent some computational operations (*e.g.*, convolution or max pooling). For convenience, we denote the number of nodes and candidate operations by B and K, respectively. In the following, we will depict the design of cell-based architectures.

Following (Pham et al., 2018; Liu et al., 2019), a cell-based architecture consists of two input nodes, $B-3$ intermediate nodes, and one output node. The input nodes denote the outputs of the nearest two cells in front of the current one. The output node concatenates the outputs of all the intermediates to produce the final output of the cell. In the DAG, each intermediate node also takes two previous nodes in this cell as inputs. In this sense, there are $2 \times (B-3)$ edges in the DAG and we will determine which operation should be applied out of K candidate operations. Based on the design of cell-based architecture, we will analyze the effect of both the number of nodes N and the number of candidate operations K on the size of search space.

First, we calculate the number of cell-based topological structures while ignoring the type of edges (*i.e.*, operations). Since each intermediate node must take two previous nodes in the same cell as inputs, there are $i-1$ optional previous nodes for the i-th intermediate node ($3 \leq i \leq B-1$). In this case, the i-th node has $(i-1)^2$ choices for the two inputs. To sum up, the number of the cell-based topological structure, denoted by M, can be calculated by

$$M = \prod_{i=3}^{B-1} (i-1)^2 = \big((B-2)!\big)^2. \tag{A}$$

Then, we calculate the number of all the possible variants w.r.t a specific topological structure. For a specific topological structure, there exist many variants because the edges of each intermediate node have not determined which operation to choose. Since the cell has $B-3$ intermediate nodes and each intermediate node has two inputs from two previous nodes, there are $2 \times (B-3)$ edges that we should decide which operation should be chosen. Moreover, each edge has K operations to be chosen. Because the choices are independent, the number of all the possible variants w.r.t a specific topological structure, denoted by L, can be calculated by

$$L = K^{2(B-3)}. \tag{B}$$

In conclusion, given B nodes and K candidate operations, the size of the search space Ω becomes:

$$|\Omega| = M \times L = K^{2(B-3)} \big((B-2)!\big)^2. \tag{C}$$

From Eqn. (C), the search space can be extremely large when we have a large B or K. For example, $|\Omega| \approx 5 \times 10^{12}$ in ENAS (Pham et al., 2018) when $B = 8$ and $K = 5$, $|\Omega| \approx 2 \times 10^{11}$ in DARTS (Liu et al., 2019) when $B = 7$ and $K = 8$. Since the size of the search space is extremely large, searching for the optimal architectures in such a large search space is a difficult problem.

B. More Training Details of CNAS

Following the settings in (Liu et al., 2019), the convolutional cells have two types, namely the normal cell and the reduction cell. The normal cell keeps the spatial resolution of the feature maps. The reduction cell reduces the height and width of the feature maps by half and doubles the number of the channels of the feature maps. Each cell contains 7 nodes, including 2 input nodes, 4 intermediate nodes, and 1 output node. The output node is the concatenation of the 4 intermediate nodes. There are 8 candidate operations between two nodes, including 3×3 depthwise separable convolution, 5×5 depthwise separable convolution, 3×3 max pooling, 3×3 average pooling, 3×3 dilated convolution, 5×5 dilated convolution, identity, and none. The convolutions are applied in the order of ReLU-Conv-BN. Each depthwise separable convolution is applied twice following (Zoph et al., 2018).

We search for good architectures on CIFAR-10 and evaluate the learned architectures on both CIFAR-10 and ImageNet. Note that some of the operations do not contain any parameters. Thus, a model cannot be trained if a cell-based architecture only consists of the operations without parameters. To avoid this issue, we make the first operation in the operation sequence to have trainable parameters, *e.g.*, depthwise separable convolution or dilated convolution.

In the training, we divide the original training set of CIFAR-10 into two parts, 40% for training the super network parameters and 60% for training the controller parameters (Pham et al., 2018). We train the controller for 320 epochs in total, 40 epochs for each stage. Before adding operations at each stage, we perform the operation warmup for 20 epochs. For training the super network, we use SGD optimizer with a weight decay of 3×10^{-4} and a momentum of 0.9. The learning rate is set to 0.1. For training the controller, we use ADAM with a learning rate of 3×10^{-4} and a weight decay of 5×10^{-4}. We add the controller's sample entropy to the reward, which is weighted by 0.005.

C. More Evaluation Details on CIFAR-10 and ImageNet

In this section, we provide more details about the evaluation method. Following (Liu et al., 2019), we conduct architecture search on CIFAR-10 and evaluate the searched architectures on two benchmark datasets, namely CIFAR-10 and ImageNet.

More evaluation details on CIFAR-10. We build the final convolution network with 20 learned cells, including 18 normal cells and 2 reduction cells. The two reduction cells are put at the $1/3$ and $2/3$ depth of the network, respectively. The initial number of the channels is set to 36. Following the setting in (Liu et al., 2019), we train the convolution network for 600 epochs using the batch size of 96. The training images are padded 4 pixels on each side. Then the padded images or its horizontal flips are randomly cropped to the size of 32×32. We also use cutout (DeVries & Taylor, 2017) with a length of 16 in the data augmentation. We use the SGD optimizer with a weight decay of 3×10^{-4} and a momentum of 0.9. The learning rate starts from 0.025 and follows the cosine annealing strategy with a minimum of 0.001. Additional enhancements include path dropout (Larsson et al., 2017) of the probability of 0.2 and auxiliary towers (Szegedy et al., 2015) with a weight of 0.4.

More evaluation details on ImageNet. We consider the ***mobile setting*** to evaluate our ImageNet models. In the mobile setting, the input image size is 224×224 and the number of multiply-adds (MAdds) in the model is restricted to be less than 600M. We build the network with 14 cells, including 12 normal cells and 2 reduction cells. The locations of the two reduction cells are the same as the setting of CIFAR-10 experiments. We follow the same setting as that in DARTS (Liu et al., 2019). Specifically, we perform the same data augmentations during training, including horizontally flipping, random crops, and color jittering. During testing, we resize the input images to 256×256 and then apply a center crop to the size 224×224. The network is trained for 250 epochs with a batch size of 256. We use SGD optimizer with a weight decay of 3×10^{-5} and a momentum of 0.9. The learning rate is initialized to 0.1 and we gradually decrease it to zero. We add auxiliary loss with a weight of 0.4 after the last reduction cell. We set the path dropout (Larsson et al., 2017) to the probability of 0.2 and label smoothing (Szegedy et al., 2016) to 0.1.

References

DeVries, T. and Taylor, G. W. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

Guo, Y., Zheng, Y., Tan, M., Chen, Q., Chen, J., Zhao, P., and Huang, J. Nat: Neural architecture transformer for accurate and compact architectures. In *Advances in Neural Information Processing Systems*, pp. 737–748, 2019.

Larsson, G., Maire, M., and Shakhnarovich, G. Fractalnet: Ultra-deep neural networks without residuals. In *International Conference on Learning Representations*, 2017.

Liu, H., Simonyan, K., and Yang, Y. DARTS: Differentiable Architecture Search. In *International Conference on Learning Representations*, 2019.

Pham, H., Guan, M. Y., Zoph, B., Le, Q. V., and Dean, J. Efficient neural architecture search via parameter sharing. In *International Conference on Machine Learning*, pp. 4092–4101, 2018.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, 2015.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8697–8710, 2018.