Supplementary Materials of
Searching by Generating: Flexible and Efficient One-Shot NAS with Architecture Generator

A. More Details of Experimental Settings

A.1. Dataset

We perform all experiments based on the ImageNet dataset [4]. Same as the settings in [1][7][2], we randomly sample 50,000 images (50 images for each class) from the training set as our validation set, and the rest is kept as the training set. The original validation set is taken as the test set to measure the final performance of each model. The resolution of input images is 224×224.

A.2. Supernet Training

We train the unified supernet for 50 epochs using batch size 256 and adopt the stochastic gradient descent optimizer with a momentum of 0.9 and weight decay of 4×10^{-5}. The learning rate is decayed based on the cosine annealing strategy from initial value 0.045. We train the unified supernet with strict fairness [3] so that each operation in all sub-blocks and each expansion rate are trained fairly.

A.3. Generator Training

After supernet training, the architecture generator is trained for 50 epochs using batch size 128 by the Adam optimizer with the learning rate 0.001, momentum (0.5, 0.999), and weight decay 0. The temperature τ of Gumbel Softmax [5] in Eq. (2) is initially set to 5.0 and annealed by a factor of 0.95 for each epoch. The trade-off parameter λ in Eq. (11) is set to 0.0003 in our experiment.

B. Details of Search Space

The marco-architecture of our unified supernet is shown in Table 6.

Table 6. Macro-architecture of the search space. MBConv 3 × 3 denotes MobileNetV2 [6] block with kernel size 3. Column-C denotes the number of output channel of a block. Column-N denotes the number of the blocks. Column-S denotes the stride of the first block when stacked for multiple blocks. Column-E denotes the expansion rate of the blocks, and the tuples of three values represent the lowest value, highest value, and steps between options (low, high, steps).

Input shape	Block	C	N	S	E
$224^2 \times 3$	Conv 3×3	32	1	2	-
$112^2 \times 32$	MBConv 3×3	16	1	1	1
$112^2 \times 16$	Unified Block	32	2	2	(2, 6, 1)
$56^2 \times 32$	Unified Block	40	4	2	(2, 6, 1)
$28^2 \times 40$	Unified Block	80	4	2	(2, 6, 1)
$14^2 \times 80$	Unified Block	96	4	1	(2, 6, 1)
$14^2 \times 96$	Unified Block	192	4	2	(2, 6, 1)
$7^2 \times 192$	Unified Block	320	1	1	(2, 6, 1)
$7^2 \times 320$	Unified Block	1280	1	1	(2, 6, 1)
$7^2 \times 1280$	Avg pool	-	1	1	-
1280	FC	1000	1	-	-

C. More Details of Architecture Generator

A random prior is encoded into a one-hot format and then is reshaped into the shape of architecture parameters to be generated. The output of the architecture generator is a parameter map with size $LayerSize \times OperationNumber$. Motivated by generative adversarial networks, we reshape a random prior so that its shape is the same as the output map. We then feed it to the architecture generator, where we can apply 2D convolution with stride 1 for processing. Without carefully tuning convolution parameters, we can ensure that shape of the output map fits different search spaces. This design is to make the generator easily adapt to different settings. We have experimented with various structures for the architecture generator (e.g., fully connected) and found that convolutional layers yield reliable results.

D. More Details of Architecture Redundancy

We illustrate architecture redundancy in the left of Fig. 8 and forced sampling (FS) in the right of Fig. 8. In the four unified blocks in Fig. 8, four depthwise convolution with kernel size 3×3 and two skip connections are used in different sub-blocks. However, the three situations on the left of Fig. 8 are treated as different because of different arrange-

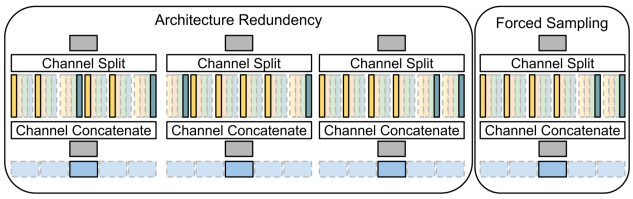

Figure 8. Illustration of architecture redundancy and forced sampling (FS).

ments. With FS, we enforce arrangement of the operations to be unique, and thus only the unified block on the right of Fig. 8 can be sampled.

E. Visualization of Searched Architectures

We visualize SGNAS-A, SGNAS-B, and SGNAS-C in Fig. 9. Besides, we also visualize the architectures searched by SGNAS under different hardware constraints in Fig. 10. It is interesting that even if the target hardware constraint is low (e.g., 280M), the expansion rate simulated by sub-blocks is still high in some layers (e.g., layer 1, layer 7, and layer 19).

F. Limitation

Careful hyperparameter tuning: In SGNAS, the overall loss function of the architecture generator is defined in Eq. (10). However, in our experiment, carefully tuning the hyperparameter λ for different datasets is required to get trade off between hardware constraints and performance.

Architecture of the architecture generator: In SGNAS, we manually design architecture of the architecture generator. But we definitely believe that there is a better architecture for the generator. It is worth further study in the future.

References

[1] Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct neural architecture search on target task and hardware. In *Proceedings of International Conference on Learning Representations*, 2019. 1

[2] Xiangxiang Chu, Bo Zhang, Jixiang Li, Qingyuan Li, and Ruijun Xu. Scarlet-nas: Bridging the gap between scalability and fairness in neural architecture search. *arXiv preprint arXiv:1908.06022*, 2019. 1

[3] Xiangxiang Chu, Bo Zhang, and Ruijun Xu. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. *arXiv preprint arXiv:1907.01845*, 2019. 1

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2009. 1

[5] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *Proceedings of International Conference on Learning Representations*, 2017. 1

[6] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 1

[7] Shan You, Tao Huang, Mingmin Yang, Fei Wang, Chen Qian, and Changshui Zhang. Greedynas: Towards fast one-shot nas with greedy supernet. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 1

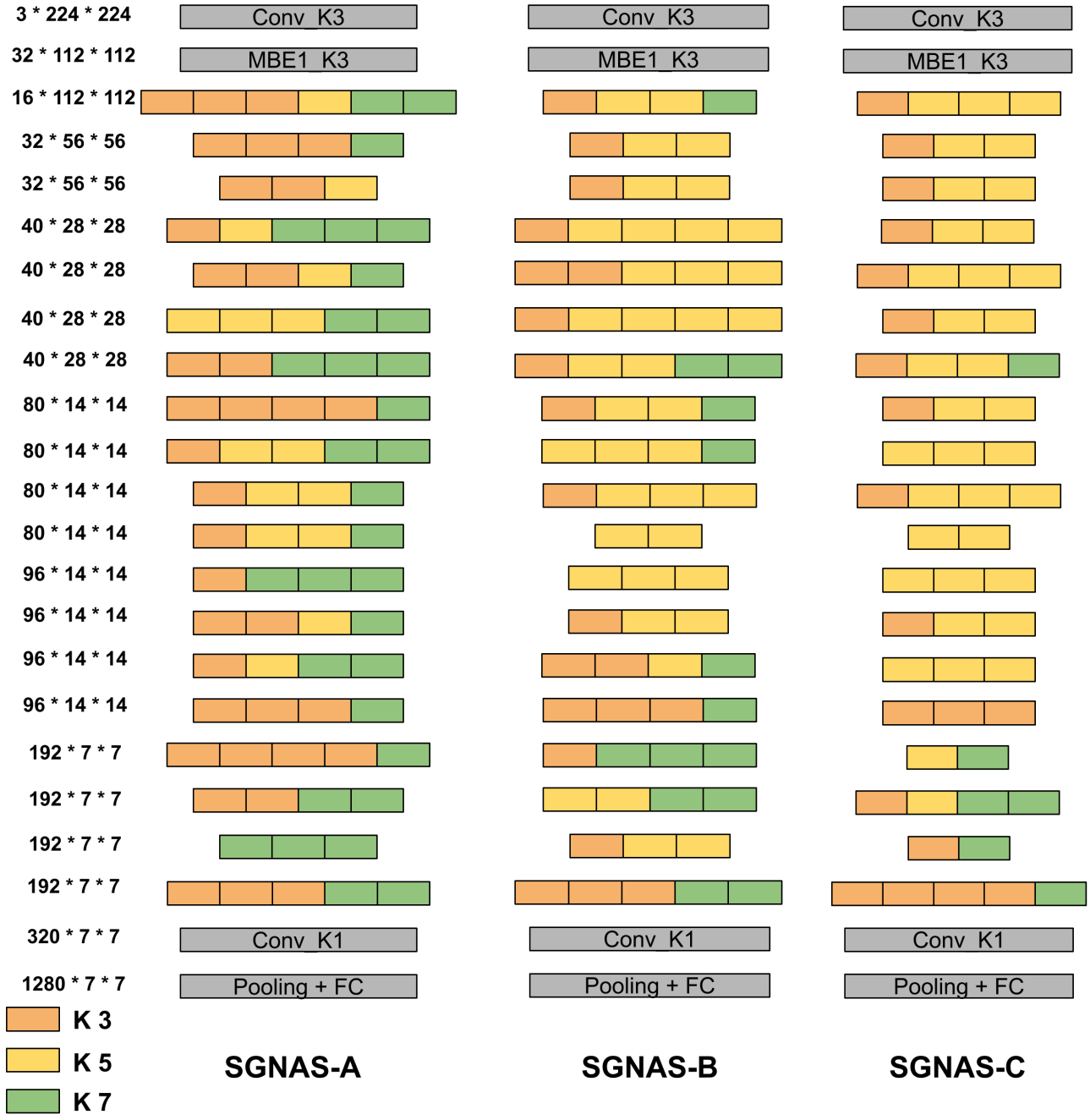

Figure 9. Visualization of the architectures searched by SGNAS (SGNAS-A, SGNAS-B, and SGNAS-C). "MBE1" denotes the mobile inverted bottleneck convolution layers with expansion rate 1. "KX" denotes depthwise convolution with the kernel size X. The gray blocks are predefined blocks before searching.

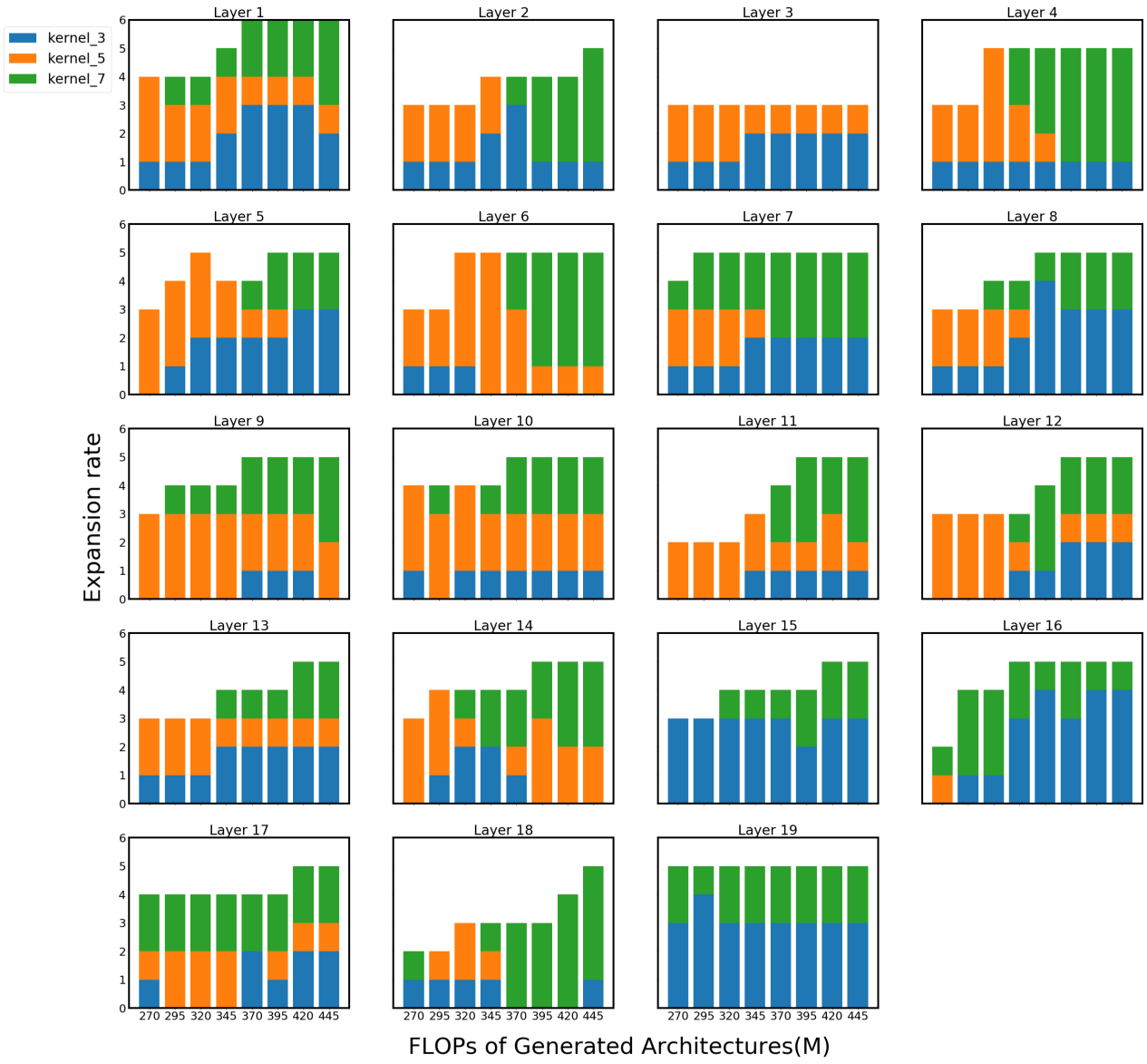

Figure 10. Visualization of the architectures search by SGNAS under different hardware constraints.