Supplementary Materials of *ScaleNet: Searching for the Model to Scale*

Jiyang Xie[1], Xiu Su[2], Shan You[3], Zhanyu Ma[1], Fei Wang[4], and Chen Qian[3]

[1] Pattern Recognition and Intelligent Systems Lab., Beijing University of Posts and Telecommunications, China. {xiejiyang2013,mazhanyu}@bupt.edu.cn
[2] The University of Sydney, Australia. xisu5992@uni.sydney.edu.au
[3] SenseTime Research Centre, China. {youshan,qianchen}@sensetime.com
[4] University of Science and Technology of China. wangfei91@mail.ustc.edu.cn

A Detailed Experimental Settings

In this section, we describe the detailed experimental settings of the proposed ScaleNet for super-supernet training, jointly base model and scaling strategies searching, model retraining, and fine-tuning tasks, respectively.

FLOPs Budgets: For both ImageNet-1k and ImageNet-100 datasets [17], we used same method to determine the FLOPs budgets for various scaling stages.

We first selected the FLOPs budget of base model by Monte Carlo simulation according to the search space in Table 1. We randomly sampled about $100,000$ paths of base model and calculated the mean FLOPs of them. We chose a multiple of 50M near the mean as the FLOPs budget f_0 of base model. For scaling stage j, we selected $2^j \cdot f_0$ as the FLOPs budget and searched the optimal scaling strategy near it.

Super-supernet Training: For ImageNet-1k, we sampled the base model from the search space in Table 1 and the scaling strategies in Table 2. We used a stochastic gradient descent (SGD) optimizer with momentum as 0.9 and weight decay as 4×10^{-5} to train the super-supernet. Initial learning was set as 0.12 with a cosine annealing strategy for $750,000$ iterations. Learning rate warm-up was also included for $3,750$ iterations, linearly increasing from zero to 0.2. We train the super-supernet with batch size as 1024. In data augmentation, we randomly resized the batches according to the resolution values in the sampled scaling strategies, with common hyper-parameters of the augmentation. Then, we resized the batch to the scaled resolutions again. The maximum scaling stage M was set as 3 in the experiments.

For ImageNet-100 dataset in ablation studies, we followed the class selection [25] and reduced the channel number of the super-supernet to a half with batch size as 256, total iterations as $150,000$ and warm-up for 375 iterations. All the other settings are same to those for ImageNet-1k experiments. The visualization of search space in the ablation studies is shown in Figure 1, which has similar trends with that for ImageNet-1k experiments.

We divided a mini validation set (50 images per class) from the training set for evaluation. The rest images in the training set were all used for super-

Table 1: The macro-structure of the search space of the base model. "n" is the number of stacked building blocks, where "$n_{\min}$" and "$n_{\max}$" are the minimum and maximum numbers, respectively. "Operation" is the type of the block. "N/A" means "not applied". "Y/N" means "using or not" in the base model of the super-supernet training. "Input" is the original resolution of input feature maps. "Channel" is the number of output channels. "Stride" is the stride of the first block. "Scale" means which dimensions of the stage need to be scaled, where "D" is depth, "W_I" is input channel number, "W_O" is output channel number, "R" is resolution, "✓" means having the part, "✗" means ignoring the one. "FC" is a fully connected layer.

Stage	n		Operation	Expand rate	SE	Input	Channel	Stride	Scale			
	$n_{\min}$	$n_{\max}$							D	W_I	W_O	R
Conv_stem	1	1	3×3 Conv.	N/A	N/A	$224 \times 224 \times 3$	$112 \times 112 \times 32$	2	✗	✗	✓	✓
Stage 1	1	1	MBConv	1	Y/N	$112 \times 112 \times 32$	$112 \times 112 \times 16$	1	✓	✓	✓	✓
Stage 2	1	4	MBConv	3/6	Y/N	$112 \times 112 \times 16$	$56 \times 56 \times 32$	2	✓	✓	✓	✓
Stage 3	1	4	MBConv	3/6	Y/N	$56 \times 56 \times 32$	$28 \times 28 \times 40$	2	✓	✓	✓	✓
Stage 4	1	4	MBConv	3/6	Y/N	$28 \times 28 \times 40$	$14 \times 14 \times 80$	2	✓	✓	✓	✓
Stage 5	1	4	MBConv	3/6	Y/N	$14 \times 14 \times 80$	$14 \times 14 \times 96$	1	✓	✓	✓	✓
Stage 6	1	4	MBConv	3/6	Y/N	$14 \times 14 \times 96$	$7 \times 7 \times 192$	2	✓	✓	✓	✓
Stage 7	1	1	MBConv	3/6	Y/N	$7 \times 7 \times 192$	$7 \times 7 \times 320$	1	✓	✓	✓	✓
Conv_out	1	1	1×1 Conv.	N/A	N/A	$7 \times 7 \times 320$	$7 \times 7 \times 1280$	1	✗	✓	✗	✓
Pooling	1	1	Global avgpool	N/A	N/A	$7 \times 7 \times 1280$	1280	N/A	✗	✗	✗	✗
Classifier	1	1	FC	N/A	N/A	1280	1000	N/A	✗	✗	✗	✗

supernet training. All the validation accuracies that we have emphasized in the paper were calculated by the mini validation set, respectively.

Jointly Base Model and Scaling Strategies Searching: We applied the proposed MCEA for the search with iteration number T as 8 and sampling number of base models for obtaining initial scaling strategies $S_j^{(0)}$ as 20.

We first conducted an initial step for obtaining the $S_j^{(0)}$ based on (7) of the main body. Then, we iteratively searched the optimal base model and scaling strategies for T iterations. In each sub-optimization process of the base model search steps, we undertook an evolution algorithm NSGA-II [6] with population size P as 50 and generation size N as 40. Meanwhile, although we can also employ the evolution algorithm for scaling strategy search steps, we directly applied a small grid search on the trained super-supernet as the search space of a scaling stage is small enough. Note that we can also use small N with large T (such as $N = 8, T = 20$), but this may be easy to fall into sub-optimal as the evolution algorithm needs to undertake crossover-mutation steps.

Model Retraining: We followed previous work [12,23] for obtaining the common training recipe. The models were trained using a RMSProp optimizer with momentum as 0.9 and weight decay as 1×10^{-5}. Initial learning was set as 0.08 with a step strategy for 300 epochs. The learning rate was increased from zero to 0.08 linearly in the first 3 epochs with batch size 1024, and then decayed to 0.97 every 2.4 epochs. In addition, exponential moving average (EMA) on weights was adopted with a decay rate 0.9999. RandAugment [5] was also introduced.

Table 2: The range of the search space of the scaling strategies. We take $M = 3$ as an example. Depth and width use the same intervals of choices. Scaling stage 0 refers to the base model.

Scaling stage	Depth / Width			Resolution		
j	Min	Step	Max	Min	Step	Max
0	1.00	0.00	1.00	1.00	0.00	1.00
1	1.04	0.04	1.16	1.00	0.07	1.14
2	1.20	0.04	1.36	1.21	0.07	1.35
3	1.40	0.04	1.64	1.43	0.07	1.57

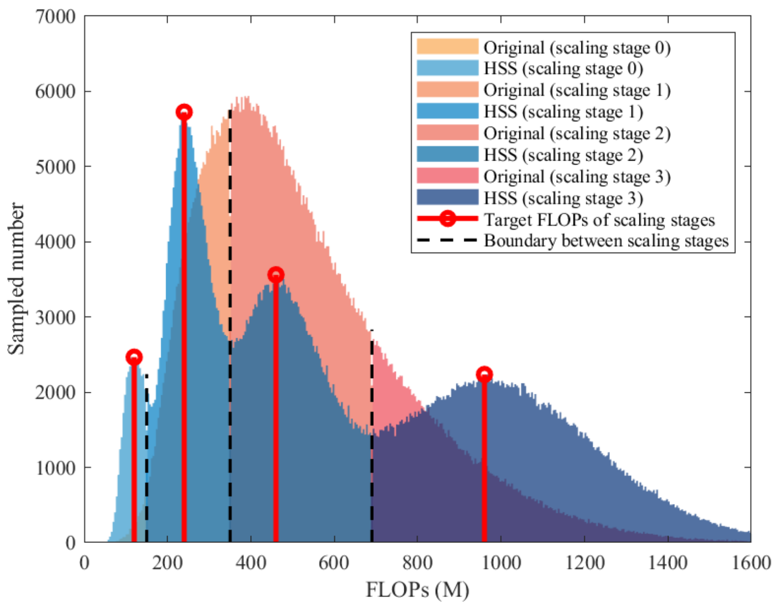


Fig. 1: Sampling distribution based on the proposed HSS, compared with that of the original uniform sampling for ablation studies (ImageNet-100). We took $750,000$ paths for each to simulate actual super-supernet training. The sampling distribution of the original uniform sampling performs as a bell-shape single-modal one, where paths cannot be fairly trained In each scaling stage (between black dashed lines). Meanwhile, paths in scaling stage 0 and 3 cannot be sufficiently trained due to the too small sampling probabilities of them. In contrary, our proposed HSS provides single-modal distributions for each scaling stage and the modes of each locate near the corresponding target FLOPs budgets.

Furthermore, for the ImageNet-100 dataset in ablation studies, we modified batch size as 256, decay rate of the EMA as 0.99, and epoch number as 200.

Fine-tuning Tasks: We followed the model retraining settings for ImageNet-1k experiments, but removed the EMA and reduced the batch size as 256.

B Detailed Settings and Definitions of Comparisons of Pearson, Spearman, and Kendall Coefficients

In Table 4 of the main body, we evaluated the proposed HSS with three coefficients, including Pearson ρ_{P} [1], Spearman ρ_{S} [29], and Kendall τ [9], compared to the original uniform sampling in [28], in order to indicate the super-supernet trained with our HSS can provide more precisely ranking for architectures. In this section, we explain the detailed settings and definitions of them, respectively.

Here, we assume the validation accuracies of the paths sampled the super-supernet should be order preserving on FLOPs same to the actual performance of those trained from scratch. This matches the assumption of the next architecture search step in the one-shot NAS that the validation accuracies of the paths can reflect on the actual performance. Thus, we utilized the three ones to evaluate the effectiveness of our HSS on the order preservation of the path performance.

We define a validation accuracy vector $\boldsymbol{ACC} = [ACC_1, \cdots, ACC_Q]^{\mathrm{T}}$ and a FLOPs vector $\boldsymbol{f} = [f_1, \cdots, f_Q]^{\mathrm{T}}$ for all the Q sampled paths, where ACC_q and f_q are both for one corresponding path. In our experiments, $Q = 6,000$. We can calculate the Pearson coefficient ρ_{P} by

$$\rho_{\mathrm{P}} = \frac{\mathrm{Cov}(\boldsymbol{ACC}, \boldsymbol{f})}{\sigma_{ACC} \cdot \sigma_f} = \frac{\mathrm{E}[(\boldsymbol{ACC} - \mu_{ACC})(\boldsymbol{f} - \mu_f)]}{\sigma_{ACC} \cdot \sigma_f}, \tag{1}$$

where $\mathrm{Cov}(\cdot,\cdot)$ is covariance function, σ_{ACC} and σ_f are the standard deviations of the two vectors, respectively, and μ_{ACC} and μ_f are the means of the two vectors, respectively.

For the Spearman coefficient ρ_{S}, as it replace the accuracy and FLOPs vectors by their rank vectors $\boldsymbol{A\tilde{C}C}$ and $\tilde{\boldsymbol{f}}$, respectively, we can compute it by

$$\rho_{\mathrm{S}} = \frac{\mathrm{Cov}(\boldsymbol{A\tilde{C}C}, \tilde{\boldsymbol{f}})}{\sigma_{A\tilde{C}C} \cdot \sigma_{\tilde{f}}}, \tag{2}$$

where $\sigma_{A\tilde{C}C}$ and $\sigma_{\tilde{f}}$ are the standard deviations of the two rank vectors, respectively. As all the ranks are distinct integers, (2) can be transferred to

$$\rho_{\mathrm{S}} = 1 - \frac{6\sum_{q=1}^{Q}(A\tilde{C}C_q - \tilde{f}_q)^2}{Q(Q^2-1)}. \tag{3}$$

Third, the Kendall coefficient τ is defined as

$$\tau = \left| \frac{Q_{\mathrm{concordant}} - Q_{\mathrm{disconcordant}}}{Q_{\mathrm{all}}} \right|, \tag{4}$$

where

$$Q_{\mathrm{concordant}} = \#\{(ACC_{q_1} - ACC_{q_2})(f_{q_1} - f_{q_2}) > 0, q_1 < q_2\}, \tag{5}$$
$$Q_{\mathrm{disconcordant}} = \#\{(ACC_{q_1} - ACC_{q_2})(f_{q_1} - f_{q_2}) \leq 0, q_1 < q_2\}, \tag{6}$$

and $Q_{\text{all}} = \frac{Q(Q-1)}{2}$.

All the three coefficients are the larger the better in the interval of $[0\%, 100\%]$. More detailed explanations of them can be found in Wiki Pedia for Pearson[5], Spearman[6], and Kendall[7] coefficients.

C Definitions of Compared Functions in Lager-scale Architecture Generalization

We have compared the proposed larger-scale architecture generalization function in the ablation studies with linear and squared functions. Here, we define the forms of the two compared functions and their optimization processes in the ablation studies.

We define the optimal scales of $M+1$ scaling stages as $\{\hat{\boldsymbol{S}}_j\}_{j=0}^{M}$. We also pre-define $\hat{\boldsymbol{S}}_0$ as $\hat{d}_0 = \hat{w}_0 = \hat{r}_0 = 1$ for the base model. These are same to those in the main body.

Linear Function. Here, we define the linear functions for the three dimensions, respectively, as

$$\begin{cases} \hat{d}_j = a_0^{(d)} \cdot j + a_1^{(d)} \\ \hat{w}_j = a_0^{(w)} \cdot j + a_1^{(w)} \\ \hat{r}_j = a_0^{(r)} \cdot j + a_1^{(r)} \end{cases}, \tag{7}$$

where a_0 and a_1 are parameters.

In order to guarantee the relation between $j=0$ and $\hat{\boldsymbol{S}}_0$, we put $\hat{d} = \hat{w} = \hat{r} = 1, j = 0$ into (7) and obtain

$$\begin{cases} a_1^{(d)} = 1 \\ a_1^{(w)} = 1 \\ a_1^{(r)} = 1 \end{cases}. \tag{8}$$

After that, we re-put (8) into (7) and obtain the linear functions for the three dimensions as

$$\begin{cases} \hat{d}_j = a_0^{(d)} \cdot j + 1 \\ \hat{w}_j = a_0^{(w)} \cdot j + 1 \\ \hat{r}_j = a_0^{(r)} \cdot j + 1 \end{cases}. \tag{9}$$

We should note that $a_0^{(d)}$, $a_0^{(w)}$, and $a_0^{(r)}$ are all constrained to be positive as the trends of the three dimensions should be monotonically increasing.

[5] `https://en.wikipedia.org/wiki/Pearson_correlation_coefficient`.
[6] `https://en.wikipedia.org/wiki/Spearman%27s_rank_correlation_coefficient`.
[7] `https://en.wikipedia.org/wiki/Kendall_rank_correlation_coefficient`.

Squared Function. Similarly, we define the squared functions for the three dimensions, respectively, as

$$\begin{cases} \hat{d}_j = a_0^{(d)} \cdot j^2 + a_1^{(d)} \cdot j + a_2^{(d)} \\ \hat{w}_j = a_0^{(w)} \cdot j^2 + a_1^{(w)} \cdot j + a_2^{(w)} \\ \hat{r}_j = a_0^{(r)} \cdot j^2 + a_1^{(r)} \cdot j + a_2^{(r)} \end{cases}, \tag{10}$$

where a_0, a_1 ,and a_2 are parameters.

In the same way, we put $\hat{d} = \hat{w} = \hat{r} = 1, j = 0$ into (10) and obtain

$$\begin{cases} a_2^{(d)} = 1 \\ a_2^{(w)} = 1 \\ a_2^{(r)} = 1 \end{cases}. \tag{11}$$

We also re-put (11) into (10) and obtain the squared functions as

$$\begin{cases} \hat{d}_j = a_0^{(d)} \cdot j^2 + a_1^{(d)} \cdot j + 1 \\ \hat{w}_j = a_0^{(w)} \cdot j^2 + a_1^{(w)} \cdot j + 1 \\ \hat{r}_j = a_0^{(r)} \cdot j^2 + a_1^{(r)} \cdot j + 1 \end{cases}, \tag{12}$$

where $a_0^{(d)}$, $a_0^{(w)}$, and $a_0^{(r)}$ are also constrained to be positive.

Optimization Process. We define the optimization objective for one dimension with a function $\mathrm{func}(\cdot; a_0, a_1)$ as

$$\begin{aligned} &\underset{a_0,a_1}{\operatorname{argmin}} \sum_{j=1}^{M} ||y_j - \mathrm{func}(j; a_0, a_1)||^2, \\ &\quad \text{s.t. } a_0 > 0, \end{aligned} \tag{13}$$

where $\mathrm{func}(\cdot; a_0, a_1)$ is selected from (9), (12), or (8) of the main body and y_j is the searched value of one dimension ($\hat{d}$, $\hat{w}$, or $\hat{r}$) in scaling stage j. We can directly optimize by stochastic gradient descent (SGD) or other optimization algorithms.

D Search Space of Base Model

We introduce the search space of base model for model scaling in Table 1. Super-supernet training will sample base models in paths according to the search space. Following previous work [19,27], we utilize mobile inverted bottleneck MB-Conv [18] with expand rates as 3 or 6. Squeeze-and-excitation network (SENet) [8] is also applied in the search space. No more than four layers are assigned in each stage for the base model. Specifically, at least one layer should be selected in each path of super-supernet training.

E Search Space of Scaling Strategies

The search space of scaling strategy is defined in Table 2. We take maximum scaling stage number $M = 3$ as an example. Note that scaling stage $j = 0$ represents the base model. As shown in Table 2, the three dimensions, *i.e.*, depth, width, and resolution, are set as one for the base model, respectively. For larger scaling stages, depth and width are selected in the same intervals, while resolution has other different ones, which are empirically designed.

For the three dimensions, we first set the center points of the intervals by Monte Carlo simulation according to the FLOPs budgets, and then obtain the ranges of the intervals based on the center points, respectively. In a dimension, we define same value of steps for each scaling stage as 0.04 for both depth and width, and 0.07 for resolution. For the former ones, a certain and suitable step is enough for the super-supernet training, while for the latter one, we empirically set them according to BigNAS [28]. Especially for width, we need to guarantee the channel numbers of different paths to uniformly change as we used the technique in [20] for width in super-supernet training.

According to the search space of base model in Table 1 and that of scaling strategy in Table 2, we can visualize the the whole search space to FLOPs by Monte Carlo simulation, as shown in Figure 3 of the main body for ImageNet-1k experiments and Figure 1 for ablation studies on ImageNet-100 (half channel numbers for each layer). Our search spaces are both in similar multi-modal forms of the histograms, instead of a simple bell-shape form of the original uniform sampling. Note that although the respective sampling distributions of each scaling stage are overlapped with contiguous ones, the paths near the boundaries between scaling stages (*i.e.*, away from the FLOPs Budgets) will be dropped in the search, as they do not satisfy the FLOPs budgets. This means that they do not affect the joint base model and scaling strategy search after the super-supernet training.

F Path Generation with Sampled Base Model and Scaling Strategy

In the super-supernet training, we specifically sample a base model architecture by uniform sampling and a scaling strategy by the proposed hierarchical sampling strategy (HSS) for each batch. Thus, we should combine them and generate a path for the super-supernet training.

For the depth, we simply copy the structure of the last non-identity block in each stage that can be scaled (See line “D” in Table 1). “Conv_stem”, “Pooling”, “Conv_out”, and “Classifier” cannot be scaled in depth. The number n_s of scaled blocks is defined as

$$n_\mathrm{s} = \lceil n \times d \rceil \,, \tag{14}$$

where n is the block number of a stage in a sampled path and d is depth coefficient.

Similarly for the width, we directly combine the added channels into each layer that can be scaled in the base model. Note that the input channel number of "Conv_stem" (three channels for RGB) and the output channel number of "Conv_out" (1280 channels same to the input channel number of the fully connected layer) are not modified (See line "W_I" and "W_O" in Table 1). The latter one is different with that of EfficientNet [23]. The scaled channel number c_{s} of a layer is defined as

$$c_{\mathrm{s}} = \lceil c \times w \rceil \,, \tag{15}$$

where c is the channel number of a layer in a sampled path and w is width coefficient.

For the resolution, we directly resize the input images before augmentations to the target size. It influences all the layers before "Pooling". The scaled resolution r_{s} of a input image is defined as

$$r_{\mathrm{s}} = \lceil 224 \times r \rceil \,, \tag{16}$$

where r is resolution coefficient.

The size of total search space in the super-supernet training is almost 2×10^{26}.

G Search Cost Computation of Referred Methods

EfficientNet [23]: As they said in the paper that they conducted a small grid search for d_1, w_1, and r_1 based on the constraint $d_1 \times w_1^2 \times r_1^2 \approx 2$, we sampled them in the interval of $[1, 2]$ with step as 0.01 and analyzed all the cases. The ones that satisfy the constraint $|d_1 \times w_1^2 \times r_1^2 - 2| \leq 0.1$ were counted and $10,285$ cases were included. For the training time of a model, we found that training an EfficientNet-B4 model (4.2G FLOPs) cost about one TPU day. Thus, training an EfficientNet-B1 model (0.7G FLOPs) may cost about 1/6 TPU days. The total search cost for the compound scaling should be $1,714$ TPU days.

Another work [26] said the lower bound of searching EfficientNet-B0 is $91,000$ TPU hours, *i.e.*, $3,792$ TPU days. However, we did not consider the part.

EfficientNet-X [13]: They undertook a two-level grid search for d_1, w_1, and r_1. We first sampled them in the interval of $[1, 2]$ with step as 0.1 and remained all the cases. Then, a smaller grid search around the best candidate with difference ranges in the interval of $[-0.1, 0.1]$ and step 0.01. Finally, we can obtain $11^3 + 21^3 - 1 = 10,591$ cases. Similar to the EfficientNet, the total search cost should be $1,756$ TPU days.

BigNAS [28]: As discussed in the paper, they train the supernet for 36 hours with 64 TPUv3. Here, the model sized from 200M to $2,000$M FLOPs and they searched from 200M to $1,000$M FLOPs. Thus, for searching a 10G FLOPs architecture, the supernet should size from 200M to 20G FLOPs, *i.e.*, $100\times$ than before. The supernet training time should be also enlarged to $10\times$ as $36 \times 64 \times 10 = 23,040$ TPU hours, *i.e.*, 960 TPU days. Although the paper did not mention their cost on searching architectures, we assumed their cost ratio of supernet training and searching is same to ours and approximated their searching cost as $960 \times (106/379) = 268$ TPU days.

Table 3: The searched and generalized scaling strategies from ScaleNet-S0 to ScaleNet-S5.

Scaling stage	Depth	Width	Resolution
0	1.000	1.000	1.000
1	1.080	1.040	1.140
2	1.360	1.200	1.355
3	1.480	1.400	1.580
4	1.653	1.534	2.042
5	1.848	1.688	2.378

FBNetV2 [26]: As shown in the paper, the authors searched the FBNetV2-L1 model (0.33M FLOPs) with total 0.6k GPU hours. We linearly expand the number by FLOPs and obtain $39,182$ GPU hours, *i.e.*, $1,633$ GPU days.

OFA [3]: They first searched a large architecture (40 GPU hours), where FLOPs budget was not mentioned in the paper and we assumed as 600M FLOPs, then trained the model for enough time ($1,200$ GPU hours), and searched target ones from the large one by progressive shrinking and fine-tuning (75×40 GPU hours). For searching and training a 10G FLOPs large model, they need $(1,200+40) \times 10,000/600 = 20,667$ GPU hours. Then, searching the five scaling stages needs at least $75 \times 40 \times 5 = 15,000$ GPU hours. The total time cost should be $20,667 + 15,000 = 35,667$ GPU hours, *i.e.*, $1,486$ GPU days.

MnasNet [22]: Their total search cost is $91,000$ TPU hours for 388M FLOPs budget. Transferring to our total FLOPs budgets as $10,000 + 6,000 + 3,000 + 1,500 + 800 + 350 = 21,650$M FLOPs, their search cost should be $91,000 \times 21,650/388 = 5,077,706$ TPU hours, *i.e.*, $211,571$ TPU days.

Speed comparisons of Different Devices: We have reviewed the computation speed of different devices, including V100 GPU and TPUv2/3, which were used in previous work [3,13,22,23,26,28] or this paper. TPUv3 is $2.7\times$ faster than TPUv2[8]. The speed of a TPUv2 core is similar to that of one V100[9]. Combining all of them, we can obtain the training speed ratios as a TPUv3 core : a TPUv2 core : a V100 = 2.7 : 1 : 1.

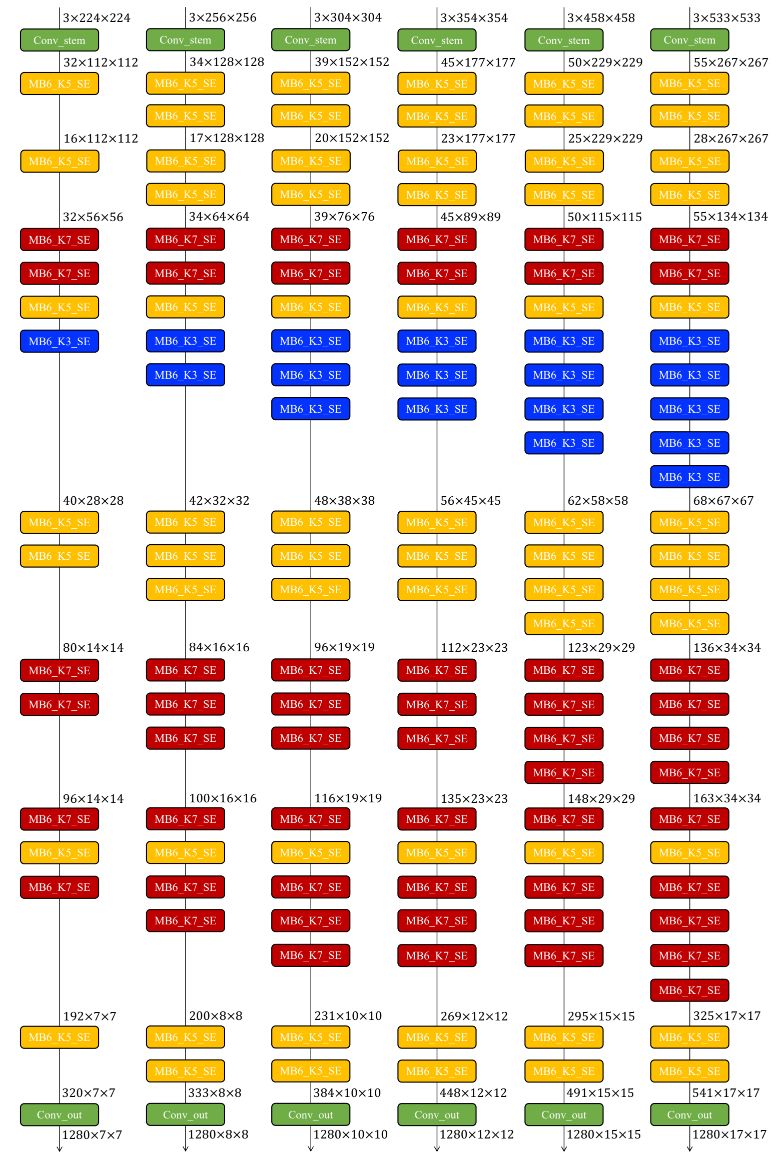
Note that we only discuss the search cost, as we assume all the methods require same cost on retraining, except for the BigNAS, which does not need to retrain the searched architectures. Retraining needs about 280 GPU days on V100 and the ScaleNet totally costs 765 GPU days, which is much smaller than $1,228$ TPU days on TPUv3 of the BigNAS.

H Visualizations of Search Architectures

Table 3 shows the searched and generalized scaling strategies of the proposed ScaleNet for the ImageNet-1k experiments. Moreover, Figure 2 shows the corre-

[8] See `https://www.linleygroup.com/newsletters/newsletter_detail.php?num=6203&year=2020&tag=3`.

[9] See page 24 in `https://storage.googleapis.com/nexttpu/index.html`.



(a) ScaleNet-S0 (b) ScaleNet-S1 (c) ScaleNet-S2 (d) ScaleNet-S3 (e) ScaleNet-S4 (f) ScaleNet-S5

Fig. 2: Visualizations of searched and generalized architectures from ScaleNet-S0 to ScaleNet-S5. "MBe_Kk_SE" means the block uses $k \times k$ depth-wise convolution as the intermediate convolution with expand rate as e and SE block. The architectures are split into stages (not scaling stages) by the resolutions. All the downsampling blocks are employed in the first layer of each stage.

Table 4: Detailed comparisons between our ScaleNet and EfficientNet. "re-impl" means we reimplemented the model with our settings. "$*$" used our scaling strategies with EfficientNet-B0. The best results are highlighted in **bold**.

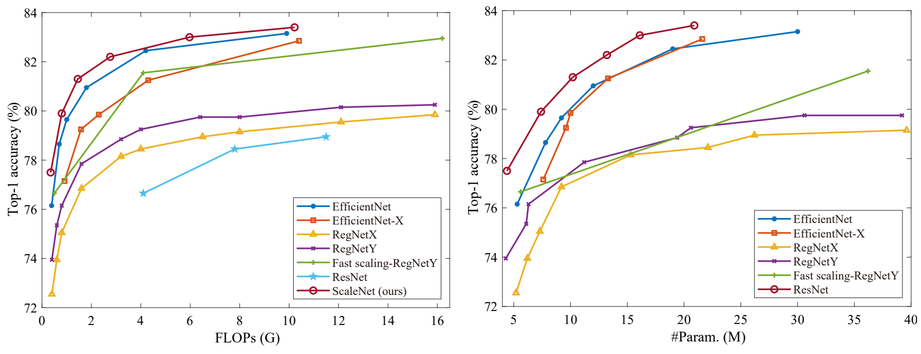
Model	Top-1 acc. (%)	Top-5 acc. (%)	#Params (M)	FLOPs (G)	Resolution
EfficientNet-B1 [23]	78.8	94.4	7.8	0.70	240×240
EfficientNet-B1 (re-impl)	78.7	94.1	7.8	0.70	240×240
EfficientNet-S1*	79.2	94.6	8.3	0.79	256×256
ScaleNet-S1	**79.9**	**94.8**	**7.4**	**0.80**	**256×256**
EfficientNet-B2 [23]	79.8	94.9	9.2	1.00	260×260
EfficientNet-B2 (re-impl)	80.4	95.1	9.2	1.00	260×260
EfficientNet-S2*	80.8	95.4	11.8	1.58	304×304
ScaleNet-S2	**81.3**	**95.6**	**10.2**	**1.45**	**304×304**
EfficientNet-B3 [23]	81.1	95.5	12.0	1.80	300×300
EfficientNet-B3 (re-impl)	81.1	95.4	12.0	1.80	300×300
EfficientNet-S3*	82.1	95.8	15.4	2.94	354×354
ScaleNet-S3	**82.2**	**95.9**	**13.2**	**2.76**	**354×354**
EfficientNet-B4 [23]	82.6	96.3	19.0	4.20	380×380
EfficientNet-B4 (re-impl)	82.6	96.3	19.0	4.20	380×380
EfficientNet-S4*	82.8	96.3	19.5	6.34	458×458
ScaleNet-S4	**83.2**	**96.6**	**16.1**	**5.97**	**458×458**
EfficientNet-B5 [23]	83.3	96.7	30.0	9.90	456×456
EfficientNet-B5 (re-impl)	83.2	96.7	30.0	9.90	456×456
EfficientNet-S5*	83.2	96.4	25.8	11.42	533×533
ScaleNet-S5	**83.7**	**97.1**	**20.9**	**10.22**	**533×533**

sponding searched and generalized architectures. Here, we find that they have the following three properties:

– Regarding kernels of convolutional layers, the base model usually prefers to use more large kernels (5×5 or 7×7 convolution layers) and more 7×7 kernels are applied in the deeper layers.
– With different FLOPs budgets, two architectures may have same numbers of layers, but with distinct width and resolution, respectively. As different values of depth can respond to a same architecture.
– Regarding resolutions of various scaling stages, the ScaleNet tends to use higher resolutions than those in [7,23] to improve the performance. They can also reduce the numbers of parameters under certain FLOPs budgets to some extent.

I Discussion of Maximum Scaling Stage in Searching

In this section, we discuss the maximum scaling stage issue in the joint search of base model and scaling strategy. As discussed in the ablation studies, searching on more scaling stages can obtain better scaling strategies. This is given that directly searching the scaling strategy on a scaling stage should have better performance than generalizing it. Although we applied the maximum scaling



(a) FLOPs versus top-1 accuracy. (b) #Param. versus top-1 accuracy.

Fig. 3: Comparisons between our ScaleNet and other baseline methods with on ImageNet-1k dataset. Both (a) FLOPs and (b) number of parameters (#Param.) are compared versus top-1 accuracy. Note that we do not illustrate ResNet in (b) as it has too many parameters.

stage $M = 3$ and generalized the model on scaling stage 4 in the ablation studies, using $M = 4$ and searching the fourth scaling stage may obtain a better one.

However, larger M implies consuming more resources, for example, training the super-supernet under current settings on ImageNet-1k needs 32 V100 cards with automatic mixed precision (AMP) or 64 V100 cards with full precision. M influences the computational complexity in both super-supernet training and searching. It is approximately exponential to the computational complexity (the computational complexity is proportional to FLOPs). However, with its increase, performance improvement of the generalized models should be approximately logarithmically increased as in Table 3. As current experimental results have reached state-of-the-art performance, utilizing double resources to training the super-supernet and searching with one more scaling stage and achieving 0.5% top-1 accuracy improvement on ImageNet-1k (similar to s4 models in Table 3 of the main body. We also trained an s5 model and found that its top-1 accuracy is 90.74%, slightly higher than 90.46% of s4 model.) between a directly searched one and a generalized one may not be necessary. Since the performance of models tends to saturate with the increase of model scales, in practice a moderate M should be favored for better and efficient search. It can be theoretically set as infinite, but a too large value of it is worthless. Thus, we only set M as three in the experiments.

J Comparisons with EfficientNet

Table 4 shows detailed comparisons between our ScaleNet and EfficientNet [23]. With same experimental settings, the searched and generalized models of our

Table 5: Statistics of five fine-tuning datasets. "#Training", "#Test", and "#Class" are numbers of training samples, test samples, and classes, respectively.

Dataset	#Training	#Test	#Class
FGVC Aircraft [15]	6,667	3,333	100
Stanford Cars [10]	8,144	8,041	196
Food-101 [2]	75,750	25,250	101
CIFAR-10 [11]	50,000	10,000	10
CIFAR-100 [11]	50,000	10,000	100

ScaleNet can significantly outperform the re-implemented EfficientNet models in each scaling stage with similar FLOPs and numbers of parameters, respectively. As our models were implemented with larger resolutions than EfficientNet ones, we can slightly reduce the numbers of parameters for each.

Meanwhile, we conducted a group of experiments on our scaling strategies with EfficientNet-B0 as base model, namely EfficientNet-S1 to EfficientNet-S5. Compared with the baseline models, they can achieve remarkable or marginal performance improvement in various scaling stages. This means our obtained scaling strategies is better than those of the baseline. Furthermore, our results can also surpass those of them, which demonstrates our searched base model is more suitable for scaling than the EfficientNet-B0.

To intuitively compare the performance of our ScaleNet with other baseline methods, we visualize the top-1 accuracies of our ScaleNet *w.r.t.* FLOPs and number of parameters, respectively, in Figure 3. As shown in Figure 3(a), the performance of the searched and generalized models can outperform all the others as an upper bound. Similarly in Figure 3(b), our models also perform best among the others with a larger margin on accuracy.

K More Comparisons on Fine-tuning Tasks

The statistics of five fine-tuning datasets are shown in Table 5, including numbers of training samples, test samples, and classes. Table 6 shows experimental results on the datasets. On FGVC Aircraft, Stanford Cars, and Food-101, we trained ScaleNet-S3/4/5 with our ImageNet-pretrained models and obtained state-of-the-art performance. Furthermore, on the other two CIFAR datasets, we can also achieve superior accuracies compared with the baseline models.

L Definition Difference between "Stage" and "Scaling Stage"

"Stage" and "scaling stage" are in two different dimensions. (1) "Stage" corresponds to depth in one model. As shown in Figure 2, a model can be divided

Table 6: Comparisons with other state-of-the-art methods in five fine-tuning tasks. Top-1 accuracies (%), FLOPs (G), numbers of parameters (#Param., M) are reported. The best results on each dataset are highlighted in **bold**.

Dataset	Model	Top-1	FLOPs	#Param.
FGVC Aircraft	EfficientNet-B3 [23]	90.7	1.80	10
	NAT-M4 [14]	90.8	0.58	5
	Inception-v4 [21]	90.9	13.00	41
	ScaleNet-S3 (ours)	91.4	2.76	11
	ScaleNet-S4 (ours)	92.8	5.97	14
	ScaleNet-S5 (ours)	**92.9**	**10.22**	**19**
Stanford Cars	NAT-M4 [14]	92.9	0.37	6
	Inception-v4 [21]	93.4	13.00	41
	EfficientNet-B3 [23]	93.6	1.80	10
	EfficientNetV2-S [24]	93.8	8.80	24
	EfficientNet-B7 [23]	94.7	37.00	64
	DAT [16]	94.8	-	-
	ScaleNet-S3 (ours)	94.4	2.76	11
	ScaleNet-S4 (ours)	95.0	5.97	14
	ScaleNet-S5 (ours)	**95.1**	**10.22**	**19**
Food-101	NAT-M4 [14]	89.4	0.36	5
	Inception-v4 [21]	90.8	13.00	41
	EfficientNet-B4 [23]	91.5	4.20	17
	ScaleNet-S3 (ours)	91.2	2.76	11
	ScaleNet-S4 (ours)	92.0	5.97	14
	ScaleNet-S5 (ours)	**92.2**	**10.22**	**19**
CIFAR-10	Proxyless-G+c/o [4]	97.2	-	6
	NASNet-A [30]	98.0	42.00	85
	EfficientNet-B0 [23]	98.1	0.39	4
	NAT-M3 [14]	98.2	0.39	6
	ScaleNet-S0 (ours)	**98.3**	**0.35**	**3**
CIFAR-100	NASNet-A [30]	87.5	42.00	85
	NAT-M3 [14]	87.7	0.49	8
	EfficientNet-B0 [23]	88.1	0.39	4
	ScaleNet-S0 (ours)	**88.4**	**0.35**	**3**

into seven stages with different channel numbers. (2) "Scaling stage" is a holistic concept that refers to various models with different FLOPs budgets, *i.e.*, the subfigures in Figure 2.

References

1. Benesty, J., Chen, J., Huang, Y., Cohen, I.: Pearson Correlation Coefficient, pp. 1–4. Springer Berlin Heidelberg (2009)
2. Bossard, L., Guillaumin, M., Van Gool, L.: Food-101 – mining discriminative components with random forests. In: ECCV. pp. 446–461 (2014)
3. Cai, H., Gan, C., Wang, T., Zhang, Z., Han, S.: Once-for-All: Train one network and specialize it for efficient deployment. In: ICLR (2020)
4. Cai, H., Zhu, L., Han, S.: ProxylessNAS: Direct neural architecture search on target task and hardware. In: ICLR (2019)
5. Cubuk, E.D., Zoph, B., Shlens, J., Le, Q.: Randaugment: Practical automated data augmentation with a reduced search space. In: NeurIPS. vol. 33, pp. 18613–18624 (2020)
6. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.A.M.T.: A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6**(2), 182–197 (2002)
7. Dollar, P., Singh, M., Girshick, R.: Fast and accurate model scaling. In: CVPR. pp. 924–932 (2021)
8. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: CVPR (2018)
9. Kendall, M.G.: A new measure of rank correlation. Biometrika **30**(1/2), 81–93 (1938)
10. Krause, J., Stark, M., Deng, J., Fei-Fei, L.: 3D object representations for fine-grained categorization. In: CVPRW. pp. 554–561 (2013)
11. Krizhevsky, A.: Learning multiple layers of features from tiny images. techreport, CIFAR (2009)
12. Li, C., Peng, J., Yuan, L., Wang, G., Liang, X., Lin, L., Chang, X.: Block-wisely supervised neural architecture search with knowledge distillation. In: CVPR (2020)
13. Li, S., Tan, M., Pang, R., Li, A., Cheng, L., Le, Q.V., Jouppi, N.P.: Searching for fast model families on datacenter accelerators. In: CVPR. pp. 8085–8095 (2021)
14. Lu, Z., Sreekumar, G., Goodman, E., Banzhaf, W., Deb, K., Boddeti, V.N.: Neural architecture transfer. IEEE Transactions on Pattern Analysis and Machine Intelligence **43**(9), 2971—-2989 (2021)
15. Maji, S., Rahtu, E., Kannala, J., Blaschko, M., Vedaldi, A.: Fine-grained visual classification of aircraft. ArXiv preprint, arXiv:1306.5151 (2013)
16. Ngiam, J., Peng, D., Vasudevan, V., Kornblith, S., Le, Q.V., Pang, R.: Domain adaptive transfer learning with specialist models. ArXiv preprint, arXiv:1811.07056 (2018)
17. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet large scale visual recognition challenge. IJCV **115**(3), 211–252 (2015)
18. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: MobileNetV2: Inverted residuals and linear bottlenecks. In: CVPR (2018)
19. Su, X., Huang, T., Li, Y., You, S., Wang, F., Qian, C., Zhang, C., Xu, C.: Prioritized architecture sampling with Monto-Carlo tree search. In: CVPR. pp. 10968–10977 (2021)
20. Su, X., You, S., Wang, F., Qian, C., Zhang, C., Xu, C.: Bcnet: Searching for network width with bilaterally coupled network. In: CVPR. pp. 2175–2184 (2021)
21. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.A.: Inception-v4, Inception-ResNet and the impact of residual connections on learning. In: AAAI (2017)

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)
23. Tan, M., Le, Q.: EfficientNet: Rethinking model scaling for convolutional neural networks. In: ICML. pp. 6105–6114 (2019)
24. Tan, M., Le, Q.V.: EfficientNetV2: Smaller models and faster training. ArXiv preprint, arXiv:2104.00298 (2021)
25. Tian, Y., Krishnan, D., Isola, P.: Contrastive multiview coding. In: ECCV. pp. 776–794 (2020)
26. Wan, A., Dai, X., Zhang, P., He, Z., Tian, Y., Xie, S., Wu, B., Yu, M., Xu, T., Chen, K., Vajda, P., Gonzalez, J.E.: FBNetV2: Differentiable neural architecture search for spatial and channel dimensions. In: CVPR (2020)
27. You, S., Huang, T., Yang, M., Wang, F., Qian, C., Zhang, C.: GreedyNAS: Towards fast one-shot nas with greedy supernet. In: CVPR (2020)
28. Yu, J., Jin, P., Liu, H., Bender, G., Kindermans, P.J., Tan, M., Huang, T., Song, X., Pang, R., Le, Q.: BigNAS: Scaling up neural architecture search with big single-stage models. In: ECCV. pp. 702–717 (2020)
29. Zar, J.H.: Spearman rank correlation. Encyclopedia of biostatistics **7** (2005)
30. Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V.: Learning transferable architectures for scalable image recognition. In: CVPR (2018)