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

Abstract. Recently, community has paid increasing attention on model scaling and contributed to developing a model family with a wide spectrum of scales. Current methods either simply resort to a one-shot NAS manner to construct a non-structural and non-scalable model family or rely on a manual yet fixed scaling strategy to scale an unnecessarily best base model. In this paper, we bridge both two components and propose ScaleNet to jointly search base model and scaling strategy so that the scaled large model can have more promising performance. Concretely, we design a super-supernet to embody models with different spectrum of sizes (*e.g.*, FLOPs). Then, the scaling strategy can be learned interactively with the base model via a Markov chain-based evolution algorithm and generalized to develop even larger models. To obtain a decent super-supernet, we design a hierarchical sampling strategy to enhance its training sufficiency and alleviate the disturbance. Experimental results show our scaled networks enjoy significant performance superiority on various FLOPs, but with at least 2.53× reduction on search cost. Codes are available at https://github.com/luminolx/ScaleNet.

Keywords: Neural architecture search (NAS), model scaling, hierarchical sampling strategy, Markov chain-based evolution algorithm

1 Introduction

Convolutional neural networks (CNNs) have achieved great performance in computer vision with various model architectures [4,5,8,11,12,21,34,35,36,37,40,42] proposed for better feature extraction abilities. Previous work [22,23,24,25,39] usually focused on how to automatically design a model architecture under a certain resource budget (*e.g.*, floating-point operations per second, FLOPs) with neural architecture search (NAS) algorithms and gained significant improvements. However, due to different levels of budgets which may occur in various applications, multi-scale architectures should be considered in practice and

* This work was supported in part by National Natural Science Foundation of China (NSFC) No. 61922015, U19B2036, and in part by Beijing Natural Science Foundation Project No. Z200002.
** Corresponding author.

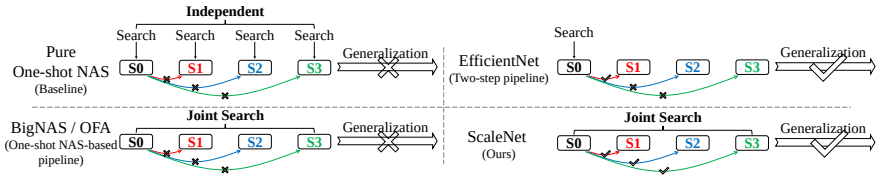


Fig. 1: Comparisons of different methods to generate a model family S0-S3. Red, blue, and green arrows are the scaling strategy searching for scaling stage 1, 2, and 3, respectively. The pure one-shot NAS (baseline) independently searched various models without either scaling strategy modeling between scaling stages or larger-scale architecture generalization. BigNAS [41] jointly searched the model family from an entire supernet, although it obtained non-structural then non-scalable architectures. Then in EfficientNet [28], the base model (*i.e.*, S0) and the scaling strategy to S1 were only searched independently for architecture generalizing. In our ScaleNet, we combine the two components and jointly search base model and all scaling strategies to scale the model into infinite ones.

can be independently generated by the NAS. Nevertheless, typical NAS methods [22,23,24,25,39] have to search one a time for each scale, and the searching cost will be approximated linearly scaled as well [29] (see baseline in Figure 1).

In contrast, recent work [3,6,7,10,15,16,17,28,31,33,41,43] get down to paying attention on model scale and designing a model family in a more straightforward way. Two frameworks have been proposed as shown in Figure 1, including one-shot NAS-based pipeline (*e.g.*, BigNAS [41] and Once-for-All (OFA) [3]) and two-step pipeline (*e.g.*, EfficientNet [28] and EfficientNet-X [15]). The former directly designed an overcomplete one-shot supernet to embody multiple (finite) scales and searched models by NAS. However, they are difficult to extend the searched models to a larger one, since finding a specific scaling strategy that adapts all the non-structurally searched architectures is infeasible. The latter decomposed the large model generation with two steps, *i.e.*, first acquiring an optimal base model, then scaling it on three dimensions, including depth, width, and resolution, using some pre-defined strategies, *e.g.*, compound scaling [28] and fast compound scaling [7]. However, the best base model is unnecessarily optimal for scaling. How to combine the advantages of both, *i.e.*, *automatically and jointly searching the base model and scaling strategy by NAS and freely extending the scaling strategy into infinite scales*, should be carefully considered.

Different with manually designed rule-based scaling strategies [7,28,38], we propose to *directly discover* the optimal scaling strategies within base model search. One-shot NAS can search model architectures based on a trained supernet that contains all the possible architectures (so-called paths). For improving search efficiency, we apply an even-larger supernet dubbed super-supernet to embody multi-scale networks. However, a common one-shot space usually has a uni-modal distribution of FLOPs of paths under uniform sampling [9,41]. In this way, the super-supernet tends to favor the intermediate-FLOPs and cannot

accommodate all FLOPs budgets well, which will in turn hampers the optimality of searched scaling strategies. Inspired by ancestral sampling [1], we propose a hierarchical sampling strategy (HSS) that splits the search space into partitions and the sampling is implemented respectively. The search space and the sampling distribution of the super-supernet for FLOPs are carefully designed according to the budgets of various scaling stages and undertake a multi-modal form distribution.

Secondly, considering that our goal is to find a base model architecture with the strongest scaling capability (instead of the best performance) and its corresponding optimal scaling strategies, we propose a joint search for them, dubbed Markov chain-based evolution algorithm (MCEA), by iteratively and interactively optimizing both of them. After obtaining the searched scaling strategies, we model the trends of depth, width, and resolution, respectively, and generalize them to develop even larger models. We theoretically derive a group of generalization functions in the three dimensions for larger-scale architectures, with which moderate performance can be actually achieved.

The contributions of this paper are four-fold:

- We propose ScaleNet to jointly search the base model and a group of the scaling strategies based on one-shot NAS framework. The scaling strategies of larger scales are generalized by the searched ones with our theoretically derived generalization functions.
- We carefully design the search space and a multi-modal distribution for FLOPs budgets for hierarchical sampling strategy (HSS) in the one-shot NAS-based scaling search algorithm to enhance the training sufficiency of paths in super-supernet.
- We propose a joint search algorithm for both the base model and the scaling strategies, namely Markov chain-based evolution algorithm (MCEA), by iteratively and interactively optimizing both of them.
- Experimental results show that the searched architectures by the proposed ScaleNet with various FLOPs budgets can outperform the referred methods on various datasets including ImageNet-1k. Meanwhile, search time can be significantly reduced at least 2.53×.

2 Related Work

2.1 One-shot NAS-based Model Family Searching

FBNets [6,31,33] optimized CNN architectures for mobile devices and generated a family of models in order to avoid training individual architectures separately and reduce resource consuming. Cai *et al.* [3] proposed to train a once-for-all (OFA) model that supports diverse architectural settings by decoupling training and search, in order to reduce the cost. BigNAS [41] challenged the conventional pipeline that post-processing of model weights is necessary to achieve good performance and constructed big single-stage models without extra retraining or

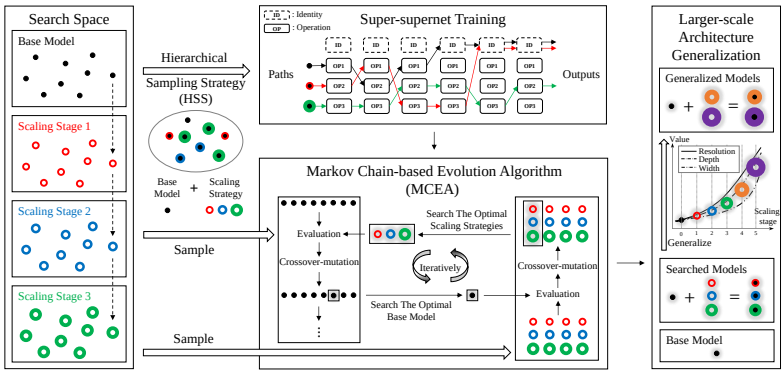


Fig. 2: Framework of the proposed ScaleNet. Based on the carefully designed search space with multiple scaling stage (the left box), we apply the proposed hierarchical sampling strategy (HSS) for sampling paths (one path is generated by a base model and a scaling strategy) in one-shot super-supernet training (the upper box). Then, we utilize the proposed Markov chain-based evolution algorithm (MCEA) to iteratively and iteractively search the optimal base model and scaling strategies (the lower box). In each iteration, an evolution procedure with crossover-mutation and evaluation is undertaken for searching the optimal base model or scaling strategies based on the search space. Finally, after obtaining the optimal ones, we generalize them to larger-scale architectures by the estimations of the trends of depth, width, and resolution, respectively (the right box). All the obtained architectures will be applied for retraining and inference.

post-processing. However, the main drawback of these methods is that they only searched for a model family by training a joint or even a group of independent supernet(s), but did not analyze the structural relationship and explicit scaling strategies between the architectures with different budgets in the model family. It is difficult and even infeasible to extend the scaling strategies to larger scales.

2.2 Model Scaling

Tan and Le [28] systematically studied model scaling and found that carefully balancing depth, width, and resolution of a model can lead to better performance. They proposed to empirically obtain the optimal compound scaling that effectively scales a specific base model up to gain a model family, *i.e.*, EfficientNet. Its variant versions, such as EfficientNetV2 [29] and EfficientNet-X [15], improved it in trade-off on speed and accuracy. A simple fast compound scaling strategy [7] was proposed to encourage to primarily scale model width, while scaling depth and resolution to a lesser extent for memory efficiency. Another work [16] built an greedy network enlarging method based on the reallocation of computations in order to enlarge the capacity of CNNs by improving the three dimensions on stage level. However, the aforementioned work always estimated

the optimal scaling strategy for theoretical double-FLOPs budget by a small grid search, which is computationally expensive and does not match the actual FLOPs budgets. Meanwhile, they only considered to find the strategy for the smallest scaling stage, which did not learn the dependency between larger scaling stages. Furthermore, the relation between a base model and scaling strategies did not be investigated, which means the base is not the optimal for scaling.

3 ScaleNet

Due to the drawbacks of the one-shot NAS-based model family searching and compound scaling-based model scaling, we propose to combine their advantages together and fill the gap between them. Here, ScaleNet jointly searches a base model with the strongest scaling capability and the optimal scaling strategies based on one-shot NAS framework by training a super-supernet as shown in Figure 2. The super-supernet training and joint searching procedures are carefully designed for the goal. Then, when obtaining the searched scaling strategies, we model the trends of depth, width, and resolution, respectively, and generalize to develop even larger architectures. All the searched and generalized scaling strategies will be applied for the final model family construction and training.

3.1 One-shot Joint Search Space for Model Scaling

The FLOPs budget of the base model is selected according to the mean FLOPs of the search space as shown in Figure 3 (Detailed information of the search space are shown in supplementary material). Then for various scaling stages, their FLOPs budgets are exponentially expanded by that of the base model α. As scaling strategies $\{\boldsymbol{S}_j = [d_j, w_j, r_j]\}_{j=0}^{M}$ with maximum scaling stage as M, including the change ratios of depth d_j, width w_j, and resolution r_j (real numbers that are not smaller than one), have their corresponding FLOPs budgets, respectively, we assign a scaling strategy to each base model architecture to compute the mean FLOPs and find the center point of the search space of scaling stage j according to its FLOPs budget. The detailed settings of the whole search space are elaborated in the supplementary material.

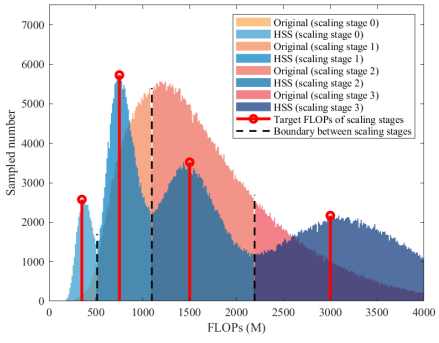


Fig. 3: Sampling distribution based on the proposed HSS, compared with that of the original uniform sampling in [41]. We took 750,000 paths for each to simulate actual super-supernet training.

3.2 Hierarchical Sampling Strategy for Super-supernet Training

Applying the original one-shot NAS framework [41,9] by utilizing uniform sampling to train the super-supernet means each operation has equal probability to be selected. It has two disadvantages: 1) the original bell-shape sampling distribution towards one specific FLOPs budget for the whole search space is not suitable for the multiple scaling stages with various budgets, where paths cannot be fairly trained in each scaling stage (see the red histogram in Figure 3); 2) the search space is almost 300× larger and the size of super-supernet is 8× larger (statistics with base model and three scaling stages trained) than before, which increase the difficulty of super-supernet training. As we need to search architectures in different scaling stages, the paths with the less selected FLOPs budgets in super-supernet training are not sufficiently trained and those in one scaling stage are not fairly trained. Here, we propose a hierarchical sampling strategy (HSS) by implementing a multi-modal sampling distribution to address the above issues (see the blue histogram in Figure 3).

As we assigned d_j, w_j, and r_j into scaling stages in the search space, we treat the target multi-modal sampling distribution $p(\boldsymbol{\alpha}, \boldsymbol{S})$ as a mixture model, *i.e.*,

$$p(\boldsymbol{\alpha}, \boldsymbol{S}) = p(\boldsymbol{\alpha}) \cdot p(\boldsymbol{S}) = p(\boldsymbol{\alpha}) \cdot \left(\sum_{j=0}^{M} \eta_j p_j(\boldsymbol{S}) \right), \tag{1}$$

where $p(\boldsymbol{\alpha})$ and $p_j(\boldsymbol{S})$ are the sampling distributions of base model $\boldsymbol{\alpha}$ and scaling stage j, respectively, and η_j is normalized component weight, $\sum_{j=0}^{M} \eta_j = 1$ and $\eta_j \geq 0$. Here, we can empirically set equal component weights, *i.e.*, $\eta_j = \frac{1}{M+1}$, or normalized combination ratios of scaling strategies in the search space.

In sampling, we apply the ancestral sampling of a probabilistic graphic model [1] that is a two-step hierarchical strategy for the scaling strategies. We firstly select a scaling stage m given the conditional distribution $p(m|\eta_1, \cdots, \eta_M)$, which is a categorical distribution as

$$p(m = j|\eta_1, \cdots, \eta_M) = \eta_j. \tag{2}$$

Then, we uniformly sample a scaling strategy $\boldsymbol{S}$ in scaling stage m. Meanwhile, a base model $\boldsymbol{\alpha}$ is sampled as well based on the original uniform sampling.

The one-shot super-supernet training process based on the proposed HSS is

$$\boldsymbol{W}^* = \underset{\boldsymbol{W}}{\operatorname{argmin}}\, loss_{\boldsymbol{\alpha}, \boldsymbol{S} \sim p(\boldsymbol{\alpha}, \boldsymbol{S})} \left(\boldsymbol{W}(\boldsymbol{\alpha}, \boldsymbol{S}); \boldsymbol{D}_{train} \right), \tag{3}$$

where $\boldsymbol{W}$ is a set of super-supernet parameters, $\boldsymbol{W}^*$ is a set of the optimal parameters, $loss$ is training loss function (commonly cross-entropy loss), $\boldsymbol{W}(\boldsymbol{\alpha}, \boldsymbol{S})$ means a path that is constructed by $\boldsymbol{\alpha}$ and $\boldsymbol{S}$, and $\boldsymbol{D}_{train}$ is training set.

3.3 Interactive Search for Base Model and Multiple Scaling Strategy

After completing the super-supernet training, we usually search both base model and a group of scaling strategies by an evolution algorithm (EA). The original

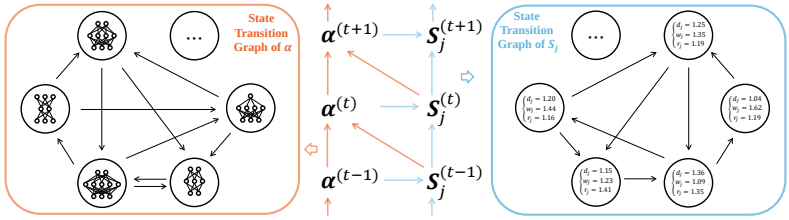


Fig. 4: The interactive search in the proposed Markov chain-based evolution algorithm (MCEA) with a coupled Markov chain. The coupled Markov chain combines both the left ((5)-(6)) and the right ((7)-(8)) ones for the interaction.

objective of the optimization is globally maximizing the weighted sum of the validation accuracies ACC of the M scaling stages [32]] as

$$\begin{aligned} &\max_{\boldsymbol{\alpha},\{\boldsymbol{S}_j\}} \left[\sum_{j=1}^{M} \pi_j ACC\left(\boldsymbol{W}^*(\boldsymbol{\alpha},\boldsymbol{S}_j);\boldsymbol{D}_{val}\right)\right], \\ &\text{s.t. } \boldsymbol{\alpha}\in\boldsymbol{\Omega}_{\boldsymbol{\alpha}}, \boldsymbol{S}_j\in\boldsymbol{\Omega}_j, \text{FLOPs}(\boldsymbol{W}^*(\boldsymbol{\alpha},\boldsymbol{S}_j)) = f_j, \end{aligned} \tag{4}$$

where π_j is the normalized weight of scaling stage j (constrained by $\sum_{j=1}^{M}\pi_j = 1$), ACC is the validation accuracy of a path, $\boldsymbol{D}_{val}$ is validation set, $\boldsymbol{\Omega}_{\boldsymbol{\alpha}}$ and $\boldsymbol{\Omega}_j$ are the search space of base model and scaling stage j, respectively, and f_j is the FLOPs budget of scaling stage j.

However, both the too large search space and constrained computational resource restrict the search. Meanwhile, searching the globally optimal group of architectures from the search space is difficult and expensive, since redundant information and noises may affect the search.

Here, inspired by Markov process, we propose a so-called Markov chain-based evolution algorithm (MCEA) with a coupled Markov chain, which iteratively and interactively optimizes $\boldsymbol{\alpha}$ and $\{\boldsymbol{S}_j\}$, to overcome the global search issue. As shown in Figure 4, with maximum iteration number T and iteration index $t = 1,\cdots,T$, we transfer the optimization problem in (4) to iteratively and interactively solving the limiting distributions $\gamma(\boldsymbol{\alpha})$ and $\gamma(\{\boldsymbol{S}_j\})$ to obtain the optimal $\boldsymbol{\alpha}^*$ and $\boldsymbol{S}_j^*$, respectively, as

$$\boldsymbol{\alpha}^* = \underset{\boldsymbol{\alpha}}{\text{argmax}}\,\gamma(\boldsymbol{\alpha}) = \text{argmax}_{\boldsymbol{\alpha}}\left[\lim_{t\to\infty} p(\boldsymbol{\alpha}^{(t)})\right], \tag{5}$$

$$p(\boldsymbol{\alpha}^{(t)}) = \sum_{\boldsymbol{\alpha}^{(t-1)}} p(\boldsymbol{\alpha}^{(t)}|\boldsymbol{\alpha}^{(t-1)}, \{\boldsymbol{S}_j\}_{j=1}^{M} = \{\boldsymbol{S}_j^{(t-1)}\}_{j=1}^{M}) p(\boldsymbol{\alpha}^{(t-1)}), \tag{6}$$

$$\boldsymbol{S}_j^* = \underset{\boldsymbol{S}_j}{\text{argmax}}\,\gamma(\boldsymbol{S}_j) = \text{argmax}_{\boldsymbol{S}_j}\left[\lim_{t\to\infty} p(\boldsymbol{S}_j^{(t)})\right], \tag{7}$$

$$p(\boldsymbol{S}_j^{(t)}) = \sum_{\boldsymbol{S}_j^{(t-1)}} p(\boldsymbol{S}_j^{(t)}|\boldsymbol{S}_j^{(t-1)}, \boldsymbol{\alpha} = \boldsymbol{\alpha}^{(t)}) p(\boldsymbol{S}_j^{(t-1)}), \tag{8}$$

where $p(\boldsymbol{\alpha}^{(t-1)})$ and $p(\boldsymbol{S}_j^{(t-1)})$ are the state probabilities of the discrete variable $\boldsymbol{\alpha}^{(t-1)}$ and $\boldsymbol{S}_j^{(t-1)}$, respectively, with the enumerable search space as state space.

$p(\boldsymbol{\alpha}^{(t)}|\boldsymbol{\alpha}^{(t-1)}, \{\boldsymbol{S}_j\}_{j=1}^{M} = \{\boldsymbol{S}_j^{(t-1)}\}_{j=1}^{M}) \propto \sum_{j=1}^{M} \pi_j ACC\left(\boldsymbol{W}^*(\boldsymbol{\alpha}, \boldsymbol{S}_j^{(t-1)}); \boldsymbol{D}_{val}\right)$ and $p(\boldsymbol{S}_j^{(t)}|\boldsymbol{S}_j^{(t)}, \boldsymbol{\alpha} = \boldsymbol{\alpha}^{(t-1)}) \propto ACC\left(\boldsymbol{W}^*(\boldsymbol{\alpha}^{(t)}, \boldsymbol{S}_j); \boldsymbol{D}_{val}\right)$ are transition matrices and are approximated implemented by the crossover-mutation process under FLOPs budgets given the obtained scaling strategies or base model, respectively. $\boldsymbol{S}_j^{(0)}$ is the initial scaling strategy of the j^{th} scaling stage, where probability $p(\boldsymbol{S}_j^{(0)}) \propto \frac{1}{K}\sum_{k=1}^{K} ACC\left(\boldsymbol{W}^*(\boldsymbol{\alpha}_k^{(0)}, \boldsymbol{S}_j); \boldsymbol{D}_{val}\right)$ is obtained by a group of randomly selected base models $\{\boldsymbol{\alpha}_k^{(0)}\}_{k=1}^{K}$ under the FLOPs budget of base model.

3.4 Larger-scale Architecture Generalization with Searched Scaling Strategies

The scaling strategies of larger scales are generalized by the searched ones. We define the optimal scales of $M+1$ scaling stages as $\{\hat{\boldsymbol{S}}_j\}_{j=0}^{M}$. We should note that we pre-define $\hat{\boldsymbol{S}}_0$ as $\hat{d}_0 = \hat{w}_0 = \hat{r}_0 = 1$ for the base model.

We argue that depth, width, and resolution should have distinct growth rates, respectively, since they perform different roles in the model scaling. Meanwhile, j is exponentially proportional to FLOPs budgets in our setting, but $\hat{\boldsymbol{S}}_j, j = 1, \cdots, M$ are under almost linear- or quadratic-level. Therefore, inspired by [28], we propose to utilize the independent regression functions for depth, width, and resolution for the larger-scale generalization with

$$\begin{cases} \hat{d}_j = a_0^{(d)} \cdot \left(\left(a_1^{(d)}\right)^j - 1\right) + 1 \\ \hat{w}_j = a_0^{(w)} \cdot \left(\left(a_1^{(w)}\right)^j - 1\right) + 1 \\ \hat{r}_j = a_0^{(r)} \cdot \left(\left(a_1^{(r)}\right)^j - 1\right) + 1 \end{cases} \tag{9}$$

to guarantee the values in $\hat{\boldsymbol{S}}_0$, where a_0 and a_1 are parameters which can be directly optimized by stochastic gradient descent (SGD) or other optimization algorithms. As we can learn different values of the parameters, respectively, the three dimensions can obtain distinct growth rates.

Derivation of Larger-scale Architecture Generalization Functions. We define FLOPs budget as f here. We can obtain the relation between f and scaling stage j as $f \propto 2^{\theta \times j}$ where $\propto$ means "proportional to", $\theta > 0$ is a parameter. As the depth $\hat{d}$, width $\hat{w}$, resolution $\hat{r}$ are positively correlated with f, we have $\theta^{(d)}$, $\theta^{(w)}$, and $\theta^{(r)}$ with $\theta = \theta^{(d)} + \theta^{(w)} + \theta^{(r)}$ for the three, and obtain

$$2^{(\theta^{(d)}+\theta^{(w)}+\theta^{(r)})\times j} \dot{\propto} \hat{d} \times \hat{w}^2 \times \hat{r}^2, \tag{10}$$

where $\dot{\propto}$ means "positively correlated with", but not "proportional to". We formulate the relation between j and $\hat{d}$ as $2^{\theta^{(d)}\times j} \dot{\propto} \hat{d}$ and introduce a linear approximation as $2^{\theta^{(d)}\times j} \approx \beta\hat{d} + \delta \Rightarrow \hat{d} \approx \frac{1}{\beta} \cdot 2^{\theta^{(d)}\times j} - \frac{\delta}{\beta}$, where β and δ are

Table 1: Comparisons with other state-of-the-art methods on ImageNet-1k dataset. Top-1 and Top-5 accuracies (%), FLOPs (G), and numbers of parameters (#Param., M) are reported. The best results are highlighted in **bold**.

Model	Top-1	Top-5	FLOPs	#Param.
FBNetV2-L1 [31]	77.2	N/A	0.33	N/A
OFA-80 [3]	76.8	93.3	0.35	6.1
GreedyNAS-A [39]	77.1	93.3	0.37	6.5
EfficientNet-B0 [28]	76.3	93.2	0.39	5.3
ScaleNet-S0 (ours)	**77.5**	**93.7**	**0.35**	**4.4**
EfficientNet-B1 [28]	78.8	94.4	0.70	7.8
OFA-200 [3]	79.0	94.5	0.78	11.0
RegNetY-800MF [19]	76.3	N/A	0.80	6.3
EfficientNet-X-B0 [15]	77.3	N/A	0.91	7.6
ScaleNet-S1 (ours)	**79.9**	**94.8**	**0.80**	**7.4**
EfficientNet-B2 [28]	79.8	94.9	1.00	9.2
BigNASModel-XL [41]	80.9	N/A	1.04	9.5
EfficientNet-X-B1 [15]	79.4	N/A	1.58	9.6
RegNetY-1.6GF [19]	78.0	N/A	1.60	11.2
ScaleNet-S2 (ours)	**81.3**	**95.6**	**1.45**	**10.2**
EfficientNet-B3 [28]	81.1	95.5	1.80	12.0
EfficientNet-X-B2 [15]	80.0	N/A	2.30	10.0
RegNetY-3.2GF [19]	79.0	N/A	3.20	19.4
RegNetY-4GF [19]	79.4	N/A	4.00	20.6
ScaleNet-S3 (ours)	**82.2**	**95.9**	**2.76**	**13.2**
RegNetY-500M$\rightarrow$4GF [7]	81.7	N/A	4.10	36.2
EfficientNet-B4 [28]	82.6	96.3	4.20	19.0
EfficientNet-X-B3 [15]	81.4	N/A	4.30	13.3
RegNetY-8GF [19]	81.7	N/A	8.00	39.2
ScaleNet-S4 (ours)	**83.2**	**96.6**	**5.97**	**16.1**
EfficientNet-B5 [28]	83.3	96.7	9.90	30.0
EfficientNet-X-B4 [15]	83.0	N/A	10.40	21.6
RegNetY-500M$\rightarrow$16GF [7]	83.1	N/A	16.20	134.8
EfficientNet-B0 $\rightarrow$16GF [7]	83.2	N/A	16.20	122.8
ScaleNet-S5 (ours)	**83.7**	**97.1**	**10.22**	**20.9**

parameters. Here, we define $a_0^{(d)} = \frac{1}{\beta},\ a_1^{(d)} = 2^{\theta^{(d)}},\ a_2^{(d)} = -\frac{\delta}{\beta}$ and obtain

$$\hat{d} = a_0^{(d)} \cdot \left(a_1^{(d)}\right)^j + a_2^{(d)}. \tag{11}$$

Note that as $a_0^{(d)}$, $a_1^{(d)}$, and $a_2^{(d)}$ are parameters, "$\approx$" can be transferred to "$=$".

Then, due to $\hat{d} = 1$ for the base model (*i.e.*, scaling stage 0), we should guarantee the relation. Thus, we put $\hat{d} = 1, j = 0$ into (11) and obtain $a_2^{(d)} = 1 - a_0^{(d)}$. We re-put it into (11) and obtain the depth function in (9) as

$$\hat{d} = a_0^{(d)} \cdot \left(a_1^{(d)}\right)^j + \left(1 - a_0^{(d)}\right) = a_0^{(d)} \cdot \left(\left(a_1^{(d)}\right)^j - 1\right) + 1. \tag{12}$$

Table 2: Performance in five fine-tuning tasks. Top-1 accuracies (%), FLOPs (G), parameter numbers (#Param., M) are reported. The best results are in **bold**.

Dataset	Model	Top-1	FLOPs	#Param.
FGVC Aircraft [18]	Inception-v4 [26]	90.9	13.00	41
	EfficientNet-B3 [28]	90.7	1.80	10
	ScaleNet-S3 (ours)	**91.4**	**2.76**	**11**
Stanford Cars [13]	Inception-v4 [26]	93.4	13.00	41
	EfficientNet-B3 [28]	93.6	1.80	10
	ScaleNet-S3 (ours)	**94.4**	**2.76**	**11**
Food-101 [2]	Inception-v4 [26]	90.8	13.00	41
	EfficientNet-B4 [28]	91.5	4.20	17
	ScaleNet-S4 (ours)	**92.0**	**5.97**	**14**
CIFAR-10 [14]	NASNet-A [44]	98.0	42.00	85
	EfficientNet-B0 [28]	98.1	0.39	4
	ScaleNet-S0 (ours)	**98.3**	**0.35**	**3**
CIFAR-100 [14]	NASNet-A [44]	87.5	42.00	85
	EfficientNet-B0 [28]	88.1	0.39	4
	ScaleNet-S0 (ours)	**88.4**	**0.35**	**3**

Similarly, we can achieve the relation between i and $\hat{w}$, $\hat{r}$, respectively, as

$$\hat{w} \approx \sqrt{\frac{1}{\beta'} \cdot 2^{\theta^{(w)} \times i} - \frac{\delta'}{\beta'}} \approx \frac{1}{\sqrt{\beta'}} \sqrt{2}^{\theta^{(w)} \times i} - \sqrt{\frac{\delta'}{\beta'}}, \tag{13}$$

$$\hat{r} \approx \sqrt{\frac{1}{\beta''} \cdot 2^{\theta^{(r)} \times i} - \frac{\delta''}{\beta''}} \approx \frac{1}{\sqrt{\beta''}} \sqrt{2}^{\theta^{(r)} \times i} - \sqrt{\frac{\delta''}{\beta''}}, \tag{14}$$

where β', δ', β'', and δ'' are parameters.

4 Experimental Results and Discussions

4.1 Performance of ScaleNet on ImageNet-1k

We conducted experiments on ImageNet-1k dataset [20] for the proposed ScaleNet with recently proposed methods. Note that we divided a mini validation set (50 images per class) from the training set for evaluation in the MCEA. The search models are named by Sj, where S0 is the base model, S1, S2, and S3 are searched by the MCEA, and S4 and S5 are the generalized ones. The FLOPs budgets are selected according to Figure 3. Detailed settings are in the supplementary material. In Table 1 with different FLOPs budgets, the searched models of our ScaleNet can achieve the best performance among those with similar FLOPs.

4.2 Transferability to Fine-tuning Tasks

In addition to the experiments on ImageNet-1k, we also transferred the searched architectures to fine-tuning tasks by fine-tuning our ImageNet-pretrained models. Experimental settings can be found in the supplementary material. Table 2 shows the transfer learning results. Ours can outperform different referred models, respectively. When applying larger models, we can gain further improvement.

Table 3: Ablation studies. Top-1 accuracies (%) of s0-s4 models on ImageNet-100 dataset are reported. In column "Sampl.", "U" and "H" are the original uniform sampling and the proposed HSS, respectively. M and T are the maximum scaling stage and the iteration of the proposed MCEA. "Val" (for T only) means the validation accuracy (%) in the MCEA. The best results are shown in **bold**.

Sampl.	M	T	Val	s0	s1	s2	s3	s4
U	3	4	N/A	84.16	86.96	87.72	89.26	90.02
H	1	4	N/A	84.06	86.30	87.93	88.86	90.30
H	2	4	N/A	84.42	86.24	88.02	89.12	90.14
H	3	1	63.58	84.18	86.34	**88.12**	88.90	89.76
H	3	2	63.38	84.20	85.86	88.00	89.44	90.18
H	3	4	**63.61**	**84.76**	**87.18**	88.10	**89.90**	90.46
H	3	6	63.59	84.44	86.42	87.80	89.54	**90.48**
H	3	8	63.53	84.50	86.48	87.64	89.30	90.36

4.3 Ablation Studies

We discuss the effect of the proposed components for the ScaleNet on ImageNet-100 dataset [20,30]. We divided a mini validation set (50 images per class) from the training set. All the following validation accuracies were calculated by the mini one. Searched models are named by sj, where s0 (120M FLOPs) is the base model, s1 (240M FLOPs), s2 (480M FLOPs), and s3 (960M FLOPs) are searched by the MCEA, and s4 (1920M FLOPs) is the generalized one. The experimental results are shown in Table 3. Detailed experimental settings and visualization of search space are in supplementary material.

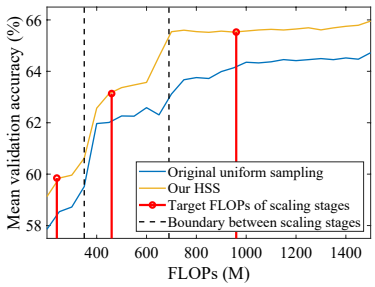


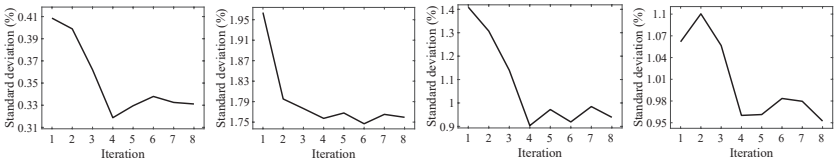
Fig. 5: Mean validation accuracies in the 4^{th} iteration of the MCEA by using original uniform sampling in [41] and our HSS, respectively. The accuracies are grouped based on FLOPs. Each non-overlapping group contains recent 50M FLOPs paths. We merely show the performance of the three scaling stages, as we only evaluate them in the MCEA, except those of the base model.

Effect of HSS: We compare the proposed HSS with the original uniform sampling in [41]. Our HSS improves the searched results with better retrained accuracies of s0-s4.

Furthermore, we illustrate the validation accuracies of paths by using both of them to evaluate the sufficiency of the super-supernet training. In Figure 5, the accuracies of our HSS are generally larger than those of the original one in addition to the FLOPs interval of $[360, 440]$, as the interval is the mode

Table 4: Performance comparison of three coefficients (%), including Pearson, Spearman, and Kendall coefficients, for validation accuracies by using original uniform sampling ("Original") in [41] and our HSS, respectively, to evaluate the sampling strategies in super-supernet training. We sampled $6,000$ paths.

Method	Pearson	Spearman	Kendall
Original	35.3	80.1	64.1
Our HSS	**73.6**	**83.9**	**66.2**



(a) Base model. (b) Scaling stage 1. (c) Scaling stage 2. (d) Scaling stage 3.

Fig. 6: Standard deviations (Stds) of validation accuracies in each iteration of the MCEA. In different scaling stages, the Stds decrease significantly in the first four iterations, while they then tend to be steady and convergent.

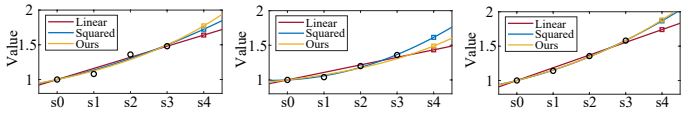
of the original uniform sampling distribution. This means the proposed HSS can improve the sufficiency of the super-supernet training.

We further analyze the Pearson, Spearman, and Kendall coefficients for validation accuracies of the two ones, respectively. All of them are the larger the better. Detailed settings are in the supplementary material. Table 4 shows the values of our HSS significantly outperform the corresponding ones of the original. Specifically, our Pearson one is more than double of the original's.

Effect of Maximum Scaling Stage in Searching: We set the maximum scaling stage M to be one, two, or three in the MCEA for searching. ScaleNet can gain better performance with larger M, which means a more suitable base model for scaling can be found. More scaling stages can achieve better performance by obtaining better base model architecture for scaling, which is a common sense. This means we do not have to validate with much larger M.

Effect of Iteration in Searching: We set the iteration T as one, two, four, six, or eight in the MCEA for searching. When increasing T from one to four, better base models and scaling strategies can be obtained with top-1 accuracies improved in most of the scaling stages. This means that larger T can improve the searched results. However, when increasing T from four to eight, similar performance can be found. This means about four iterations is enough for the search. Meanwhile, we can find that the retraining accuracies of s0-4 is relative to the validation accuracy in the MCEA, which shows the effectiveness of it.

In addition, we show the standard deviations (Stds) of validation accuracies in each iteration of the MCEA in Figure 6 to analyze their convergence. In different scaling stages, the Stds decrease significantly in the first four iterations, while



(a) Depth. (b) Width. (c) Resolution.

Method	Top-1 acc.
Linear	89.90
Squared	89.26
Ours	90.46

(d) s4 model.

Fig. 7: Comparison of larger-scale architecture generalization functions. The definitions of the compared ones are in the supplementary material. We generalized the three ones of s4 and retrained the scaled models on ImageNet-100 in (d).

Table 5: Comparisons of search cost (GPU/TPU days). Those of supernet (super-supernet) training and searching are compared, respectively. "Ratio-to-ScaleNet" is the ratio between total cost of a model to that of the proposed ScaleNet, the smaller the better. "†" means we estimated the lower-bound time. "N/A" means the work does not have the step. "*" means the work has the step but did not specifically mentioned in the paper. The best result is in **bold**.

Model	Device	Training	Searching	Total	Ratio-to-ScaleNet
MnasNet [27]†	TPUv2	N/A	211,571	211,571	436.23×
EfficientNet-X [15]†	TPUv3	N/A	> 1,765	> 1,765	> 3.64×
EfficientNet [28]†	TPUv3	N/A	> 1,714	> 1,714	> 3.53×
FBNetV2 [31]†	V100	*	*	> 1,633	> 3.37×
OFA [3]†	V100	*	*	> 1,486	> 3.06×
BigNAS [41]†	TPUv3	> 960	> 268	> 1,228	> 2.53×
ScaleNet (ours)	**V100**	**379**	**106**	**485**	**1×**

they then tend to be steady and convergent. This shows that our ScaleNet can effectively search the optimal ones and gradually minimize the Stds.

Effect of Larger-scale Architecture Generalization: We experimentally compare the proposed exponential one with commonly used polynomial functions, such as linear and squared ones. As shown in Figure 7(a)-(c), three cases can precisely fit the trends of the depth, width, and resolution, respectively. Ours function can perform with different trends in the three dimensions. For depth and resolution, ours obtains rapid increase similar to the squared one, while it achieves gradual changes for width as the linear one. The total trends of ours are various, which is similar to the conclusion in [28], but the other two always perform the uppermost or the lowest, which are unreasonable.

We also trained all the scaled s4 models with three generalized scaling strategies, respectively, shown in Figure 7(d). The proposed one can achieve the best top-1 performance as 90.46%, superior to the other two functions. This shows the effectiveness of our larger-scale architecture generalization.

4.4 Discussion of Search Cost

We discuss the efficiency of our ScaleNet, compared with a few recent strategies, including both one-shot NAS-based and two-step pipelines. We estimated the

search cost for the referred ones under our FLOPs budgets, as they applied with various FLOPs budgets. The estimations are all shown in the supplementary material. As shown in Table 5, the proposed ScaleNet can remarkably reduce the total search cost, which contains the cost of (super-)supernet training and searching. It can decrease at least 2.53× and even 436.23×. Meanwhile, the proposed ScaleNet in the two parts of cost can still significantly improve the efficiency, respectively. Note that we used V100 for our experiments, while some others utilized TPUv3, which are much better than ours. This means we can achieve a larger decrease on the total search time under same resource conditions.

4.5 Discussion of the Trend of Scaling Strategies

We discuss the trend of learned scaling strategies for different scaling stages (Figure 8) in order to promote further scaling strategy design.

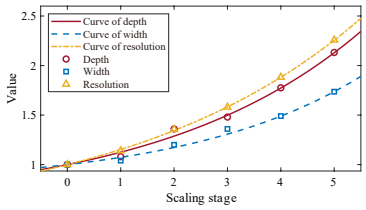


Fig. 8: Trends of depth, width, and resolution on ImageNet-1k.

- Depth, width, and resolution change in an exponential order. They have different values of their change rates among scaling stages. Depth and resolution change similarly, while that of width is slightly smaller, which is similar to EfficientNet [28].
- Their values are not completely restricted by the theoretical constraint in [28], but merely focus on the actual FLOPs budgets in each scaling stage. This means the searching process of ours is more fair.
- After obtaining the searched ones that are good enough under the corresponding FLOPs budgets, the extended ones can be precisely constructed in FLOPs of generalized architectures as well under our estimation and work well in experiments.

5 Conclusion

In this paper, we proposed ScaleNet to jointly search the base model and a group of the scaling strategies based on one-shot NAS framework. We improved the super-supernet training by the proposed HSS. Then, we jointly searched the base model and the scaling strategies by the proposed MCEA. The scaling strategies of larger scales were decently generalized by the searched ones. Experimental results show that the searched architectures by the proposed ScaleNet with various FLOPs budgets can outperform the referred methods on various datasets, including ImageNet-1k and fine-tuning tasks. Meanwhile, the searching time can be significantly reduced, compared with those one-shot NAS-based and manually designed two-step pipelines.

References

1. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer Science+Business Media LLC. (2006)
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. Chang, D., Pang, K., Zheng, Y., Ma, Z., Song, Y.Z., Guo, J.: Your "flamingo" is my "bird": Fine-grained, or not. In: CVPR. pp. 11476–11485 (2021)
5. Cheng, Z., Su, X., Wang, X., You, S., Xu, C.: Sufficient vision transformer. In: KDD (2022)
6. Dai, X., Wan, A., Zhang, P., Wu, B., He, Z., Wei, Z., Chen, K., Tian, Y., Yu, M., Vajda, P., Gonzalez, J.E.: FBNetV3: Joint architecture-recipe search using predictor pretraining. In: CVPR. pp. 16276–16285 (2021)
7. Dollar, P., Singh, M., Girshick, R.: Fast and accurate model scaling. In: CVPR. pp. 924–932 (2021)
8. Du, R., Xie, J., Ma, Z., Chang, D., Song, Y.Z., Guo, J.: Progressive learning of category-consistent multi-granularity features for fine-grained visual classification. IEEE TPAMI (2021)
9. Guo, Z., Zhang, X., Mu, H., Heng, W., Liu, Z., Wei, Y., Sun, J.: Single path one-shot neural architecture search with uniform sampling. In: ECCV. pp. 544–560 (2020)
10. Han, K., Wang, Y., Zhang, Q., Zhang, W., Xu, C., Zhang, T.: Model rubik's cube: Twisting resolution, depth and width for TinyNets. In: NeurIPS (2020)
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
12. Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q.V., Adam, H.: Searching for mobilenetV3. In: ICCV (October 2019)
13. Krause, J., Stark, M., Deng, J., Fei-Fei, L.: 3D object representations for fine-grained categorization. In: CVPRW. pp. 554–561 (2013)
14. Krizhevsky, A.: Learning multiple layers of features from tiny images. techreport, CIFAR (2009)
15. 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)
16. Liu, C., Han, K., Xiao, A., Deng, Y., Zhang, W., Xu, C., Wang, Y.: Greedy network enlarging. ArXiv preprint, arXiv:2108.00177 (2021)
17. Lou, W., Xun, L., Sabet, A., Bi, J., Hare, J., Merrett, G.V.: Dynamic-OFA: Runtime DNN architecture switching for performance scaling on heterogeneous embedded platforms. In: CVPRW. pp. 3110–3118 (2021)
18. Maji, S., Rahtu, E., Kannala, J., Blaschko, M., Vedaldi, A.: Fine-grained visual classification of aircraft. ArXiv preprint, arXiv:1306.5151 (2013)
19. Radosavovic, I., Kosaraju, R.P., Girshick, R., He, K., Dollar, P.: Designing network design spaces. In: CVPR (2020)
20. 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)
21. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: MobileNetV2: Inverted residuals and linear bottlenecks. In: CVPR (2018)

22. 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)
23. Su, X., You, S., Huang, T., Wang, F., Qian, C., Zhang, C., Xu, C.: Locally free weight sharing for network width search. ArXiv preprint, arXiv:2102.05258 (2021)
24. 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)
25. Su, X., You, S., Xie, J., Zheng, M., Wang, F., Qian, C., Zhang, C., Wang, X., Xu, C.: ViTAS: Vision transformer architecture search. ArXiv preprint, arXiv:2106.13700 (2021)
26. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.A.: Inception-v4, Inception-ResNet and the impact of residual connections on learning. In: AAAI (2017)
27. 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)
28. Tan, M., Le, Q.: EfficientNet: Rethinking model scaling for convolutional neural networks. In: ICML. pp. 6105–6114 (2019)
29. Tan, M., Le, Q.V.: EfficientNetV2: Smaller models and faster training. ArXiv preprint, arXiv:2104.00298 (2021)
30. Tian, Y., Krishnan, D., Isola, P.: Contrastive multiview coding. In: ECCV. pp. 776–794 (2020)
31. 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)
32. Wang, C., Xu, C., Yao, X., Tao, D.: Evolutionary generative adversarial networks. IEEE Transactions on Evolutionary Computation **23**(6), 921–934 (2019)
33. Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y., Keutzer, K.: FBNet: Hardware-aware efficient ConvNet design via differentiable neural architecture search. In: CVPR (2019)
34. Xie, J., Ma, Z., Chang, D., Zhang, G., Guo, J.: GPCA: A probabilistic framework for Gaussian process embedded channel attention. IEEE TPAMI (2021)
35. Xie, J., Ma, Z., Lei, J., Zhang, G., Xue, J.H., Tan, Z.H., Guo, J.: Advanced dropout: A model-free methodology for Bayesian dropout optimization. IEEE TPAMI (2021)
36. Xie, J., Ma, Z., Xue, J.H., Zhang, G., Sun, J., Zheng, Y., Guo, J.: DS-UI: Dual-supervised mixture of Gaussian mixture models for uncertainty inference in image recognition. IEEE TIP **30**, 9208–9219 (2021)
37. Xu, H., Su, X., Wang, Y., Cai, H., Cui, K., Chen, X.: Automatic bridge crack detection using a convolutional neural network. Applied Sciences **9**(14), 2867 (2019)
38. Yang, Z., Liu, D., Wang, C., Yang, J., Tao, D.: Modeling image composition for complex scene generation. ArXiv preprint, arXiv:2206.00923 (2022)
39. You, S., Huang, T., Yang, M., Wang, F., Qian, C., Zhang, C.: GreedyNAS: Towards fast one-shot nas with greedy supernet. In: CVPR (2020)
40. You, S., Xu, C., Xu, C., Tao, D.: Learning from multiple teacher networks. In: KDD. pp. 1285–1294 (2017)
41. 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)
42. Zagoruyko, S., Komodakis, N.: Wide residual networks. ArXiv preprint, arXiv:1605.07146 (2016)
43. Zhai, X., Kolesnikov, A., Houlsby, N., Beyer, L.: Scaling vision transformers. ArXiv preprint, arXiv:2106.04560 (2021)

44. Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V.: Learning transferable architectures for scalable image recognition. In: CVPR (2018)