

SCARLET-NAS: Bridging the gap Between Scalability and Fairness in Neural Architecture Search

Xiangxiang Chu¹, Bo Zhang¹ Jixiang Li¹ Qingyuan Li², Ruijun Xu¹

¹Xiaomi AI Lab, ²Xiaomi IoT

{chuxiangxiang, zhangbo11, lijixiang, liqingyuan, xuruijun}@xiaomi.com

Abstract

One-shot neural architecture search features fast training of a supernet in a single run. A pivotal issue for this weight-sharing approach is the lacking of scalability. A simple adjustment with identity block renders a scalable supernet but it arouses unstable training, which makes the subsequent model ranking unreliable. In this paper, we introduce linearly equivalent transformation to soothe training turbulence, providing with the proof that such transformed path is identical with the original one as per representational power. The overall method is named as SCARLET (SCALable supeRnet with Linearly Equivalent Transformation). We show through experiments that linearly equivalent transformations can indeed harmonize the supernet training. With an EfficientNet-like search space and a multi-objective reinforced evolutionary backend, it generates a series of competitive models: SCARLET-A achieves 76.9% top-1 accuracy on ImageNet which outperforms EfficientNet-B0 by a large margin; the shallower SCARLET-B exemplifies the proposed scalability which attains the same accuracy 76.3% as EfficientNet-B0 with much fewer FLOPs; SCARLET-C scores competitive 75.6% with comparable sizes. The models and evaluation code are released online¹.

Introduction

Neural architecture search has been recently dominated by one-shot methods (Brock et al. 2018; Bender et al. 2018; Stamoulis et al. 2019; Guo et al. 2019; Cai, Zhu, and Han 2019). Fundamentally, a supernet which incorporates the whole search space enjoys fast convergence through weight sharing. Evaluating the performance of models by picking a single path from the supernet then becomes handy. According to (Chu et al. 2019), fair training of supernet shows a remarkable improvement in model ranking. However, the scalability of a supernet is quite limited. By contrast, as pure reinforcement or evolutionary approaches train each model independently for evaluation, shallower models can also stand out if they exhibit good performance. This step is very beneficial as it achieves automatic architectural compression. To enable the similar property for the family of one-shot methods, we install identity choice blocks for network downscal-

ing, which are accompanied with *linearly equivalent transformation* as a relay for inter-block information.

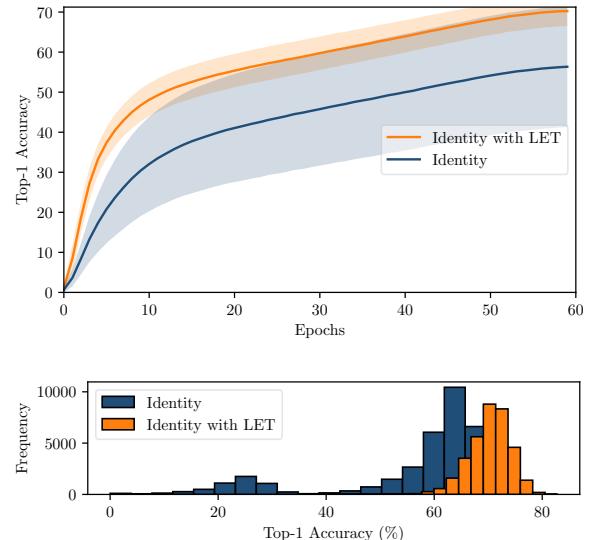


Figure 1: Training process of scalable supernets with and without linearly equivalent transformation (LET). **Top:** The accuracy of supernet with LET has much smaller standard variance. **Bottom:** Histogram of training accuracies of sampled one-shot models within last epoch.

To summarize, our main contributions are threefold.

- We uncover what is an important but neglected illness of being unscalable in previous one-shot architecture search methods.
- We present proofs on why normal identity would fail to assure fair scalability. We propose a naive adjustment with linearly equivalent transformation that distinctly improves the supernet strength.
- Our generated networks enjoy overwhelming performance compared to their counterparts especially EfficientNet-B0 (Tan and Le 2019). SCARLET-A claims a new state of the art 76.9% top-1 accuracy on ImageNet at the level of 400M multiply-adds. SCARLET-

¹<https://github.com/xiaomi-automl/SCARLET-NAS>

C also reaches competitive 75.6% with much decreased multiply-adds. More importantly, the closest model to EfficientNet-B0, SCARLET-B, which is a shallow version from our search space, distinguishes itself with 76.3%.

Related Works

One-Shot Neural Architecture Search

In one-shot approaches, a supernet is constructed to represent the whole search space, within which each path is a stand-alone model. The supernet is trained only once, child models can inherit the weights of supernet thus it is easier and faster to evaluate its performance compared with other incomplete training techniques. Notable works are (Bender et al. 2018; Stamoulis et al. 2019; Guo et al. 2019; Cai, Zhu, and Han 2019). Recent advances are concerned with training stability and mismatching accuracy range between one-shot models and stand-alone ones, FairNAS achieves better convergence and results by enforcing strict fairness constraints (Chu et al. 2019).

Scalability and Network Transformation

Given a neural network, it has been experimentally studied to scale its size up or down for various application scenarios. Common model upscaling practices include increasing depth, width, as well as input image resolution (He et al. 2016; Zagoruyko and Komodakis 2016; Huang et al. 2018). Recent work by (Tan and Le 2019) proposes an effective compound scaling method that incorporates all three with a balance achieved by grid search. Nevertheless, these methods leave parameter sharing out of the discussion, each scaled network has to be trained from scratch. Network transformation is a solution for model scaling by reusing the weights from the original structure. For instance, Net2Net by (Chen, Goodfellow, and Shlens 2015) invented two transformation schemes to pass on parameters to either wider or deeper student networks.

Review of the Supernet Training with Variable Depths

It has been unheeded about the training of scalable supernets, at least not carefully dealt with. Generally, mainstream one-shot approaches suffer from unstable training (Bender et al. 2018). This problem deteriorates when scalability is considered. Zero operations are added to skip blocks in (Cai, Zhu, and Han 2019) to have flexibility in width and depth, however, its training details are not thoroughly discussed. (Guo et al. 2019) also adopts the same search space design to draw a fair comparison. The intermediate process is also not reported, we can not decide how much difference do skip connections make.

We are motivated by (Chu et al. 2019) which displays interesting feature maps after first choice blocks. It attests that all choice blocks learn similar knowledge, even at correspondent channels. This is an important incentive for us to investigate further whether a fair training of scalable supernet is possible. We argue that ensuring fairness for common scalability practices like identity blocks is necessary.

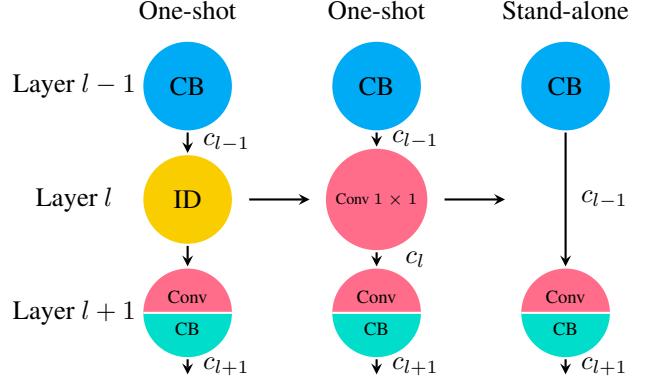


Figure 2: Identity with linearly equivalent transformation. CB: Choice Block. ID: Identity. Note c_l means the output channel size of layer l . The Conv operation in layer $l+1$ can also be FC.

As pure identity blocks are direct short paths and don't learn any information, we have to accommodate this defect by injecting a learning unit. Here we remedy the issue with 1×1 convolution without non-linear activations. It is named as linearly equivalent transformation (LET), see Figure 2. Be aware that layer $l+1$ in stand-alone models have $c_{l-1} * c_{l+1}$ parameters, while its one-shot counterparts have $c_l * c_{l+1}$.

Given an input of a chickadee² image from ImageNet, we illustrate both high-level and low-level feature maps of the trained supernet with our proposed improvements in Figure 5. Pure identity easily interferes with the training process as it causes possible channel reduction and non-congruence with other choice blocks. Note the channel size of feature map after Choice 6 in Figure 5 (a) is half of others because the previous channel size is 16, while other choice blocks output 32 channels. This effect is largely attenuated with LET. As it goes deeper, we still observe consistent high-level features. Specifically, when LET is not enforced, high-level features of deeper channels easily get blurred out, while the supernet with LET enabled continues to learn useful features in deeper channels. We will discuss it formally in the next section.

Stabilize Training by Equivalent Transformation

A critical requirement for transformation is equivalence. Thus, the transformed model behaves exactly as the original entity. To proceed, we first define the equivalence.

Definition 1. Given a valid space X , a function f is equivalent to g on X if and only if $f(x) = g(x)$ for $x \in X$, where x , $f(x)$ and $g(x)$ are tensors of any shape.

Definition 2. The equivalence of two neural networks: for models A and B with weights θ_A and θ_B , $A = B$ if and only if f_A and f_B are equivalent.

²ImageNet ID: n01592084_7680

The 2D convolution and fully-connected layer are two of the most widely used operations in deep neural networks. For the image classification task, a mini-batch of $h \times w$ images with c channels can be denoted by $I_0(b, h, w, c)$. Let A be a deep neural network with L layers and I_l be the feature maps of the l -th layer. For simplicity, we omit the batch dimension b . Hence,

$$I_l = F_l(I_{l-1}) \quad (1)$$

In general, the shape of I_l can be (h_l, w_l, c_l) or (c_l) , where the former is generated from 2D convolutions and the latter is from fully-connected operations. We denote a 2D convolution as $\text{Conv}(c_{in}, c_{out}, \text{kernel}, \text{stride})$. For convenience, we use square kernels and neglect dilation rates, considering only $\text{stride} = 1$. Other setups can be easily proven in the same manner. Here, we consider two scenarios:

- I_{l-1} and I_l have shape (c_{l-1}) and (c_l) respectively. F_l is a linear operation without activation and the first operation of F_{l+1} is a linear (fully-connected) $\text{FC}_{(c_l, m)}$,
- I_{l-1} and I_l each have shape $(h_{l-1}, w_{l-1}, c_{l-1})$ and (h_l, w_l, c_l) . F_l is a 2D convolution $\text{Conv}_{(c_{l-1}, c_l, 1, 1)}$ without activation or bias. The first operation of F_{l+1} is 2D convolution $\text{Conv}_{(c_l, m, k, 1)}$.

Note that the choice block in a layer can either be a simple or a complex block and we only require that the beginning part of it is a 2D convolution or FC. This requirement is weak enough to cover most neural networks. \square

Lemma 1. *For the first scenario, we replace F_l and $\text{FC}_{(c_l, m)}$ with an identity operation and another linear $\text{FC}_{(c_{l-1}, m)}$ to construct B from A , then we can ensure $A = B$.*

Proof. First, we copy B 's weights from A except for F_l and $\text{FC}_{(c_l, m)}$. We can make $A = B$ if we can let $\text{FC}_{(c_{l-1}, m)}$ be equivalent to the above two successive operations. For any $x \in \mathbf{R}^{c_{l-1}}$, let $W_{c_{l-1} \times c_l}$ and $W_{c_l \times m}$ denote the weight matrices of $\text{FC}_{(c_{l-1}, c_l)}$ and $\text{FC}_{(c_l, m)}$. Let $W_{c_{l-1} \times m}$ be the weight matrix of $\text{FC}_{(c_{l-1}, m)}$.

Second, we can calibrate

$$W_{c_{l-1} \times m} = W_{c_{l-1} \times c_l} W_{c_l \times m}.$$

We can make $A = B$ by combining them both. \square

Lemma 2. *For the second scenario, we substitute F_l and $\text{Conv}_{(c_l, m, k, 1)}$ with an identity operation and another 2D convolution $\text{Conv}_{(c_{l-1}, m, k, 1)}$ to construct B from A , then we can ensure $A = B$.*

Proof. First, we copy B 's weights from A except for the F_l and $\text{Conv}_{(c_l, m, k, 1)}$. The only thing to prove is that we can replace F_l and $\text{Conv}_{(c_l, m, k, 1)}$ with $\text{Conv}_{(c_{l-1}, m, k, 1)}$ equivalently. Second, We prove that any $x \in \mathbf{R}^{h, w, c_{l-1}}$, the above declaration holds. Let $W_{c_{l-1}, c_l, 1, 1}^1$ and $W_{c_l, m, k, k}^2$ be the weight matrices of F_l and $\text{Conv}_{(c_l, m, k, 1)}$. Let $W_{c_{l-1}, m, k, k}^3$ be the weight matrix of $\text{Conv}_{(c_{l-1}, m, k, 1)}$.

be the weight matrix of $\text{Conv}_{(c_{l-1}, m, k, 1)}$.

$$\begin{aligned} y &= F_l(x) \\ z &= \text{Conv}_{(c_l, m, k, 1)}(y) \\ &= \sum_{p=1}^{c_{l-1}} W_{p, c, 1, 1}^1 x(i, j, p) \\ z(i, j, c) &= \sum_{q=1}^k \sum_{p=1}^{c_l} W_{p, c, q, q}^2 y(i + q, j + q, p) \\ &= \sum_{q=1}^k \sum_{p=1}^{c_l} \sum_{u=1}^{c_{l-1}} W_{p, c, q, q}^2 (\sum_{u=1}^{c_{l-1}} W_{u, p, 1, 1}^1 x(i + q, j + q, u)) \\ &= \sum_{q=1}^k \sum_{p=1}^{c_l} \sum_{u=1}^{c_{l-1}} W_{p, c, q, q}^2 W_{u, p, 1, 1}^1 x(i + q, j + q, u) \\ &= \sum_{q=1}^k \sum_{u=1}^{c_{l-1}} W_{u, c, q, q}^3 x(i + q, j + q, u) \end{aligned}$$

We can make $A = B$ by setting

$$W_{u, c, q, q}^3 = \sum_{p=1}^{c_l} W_{p, c, q, q}^2 W_{u, p, 1, 1}^1.$$

\square

Since we don't search the number of channels, we replace the choice identity with 1×1 convolution without bias or activation for convolution networks and FC layer without activation when the identity is needed between FC layers. This procedure is illustrated in Figure 2. The left-most architecture is the original and commonly used version. The middle one is its equivalent version and the right-most version is the architecture for stand-alone training. Attention should be paid for training stand-alone models. The number of feature maps usually increases with depth. For instance, in Figure 2, $c_{l+1} \geq c_l \geq c_{l-1}$, we should adjust the input channel number to make convolution works.

Experiment Setup

Dataset

We perform the search directly on ImageNet 1k dataset (Deng et al. 2009) and randomly select 50k images from the training set as a validation set. The original validation set is used as the test set to report accuracy.

Search Space

We consider two search spaces in this paper. The difference lies in whether the squeeze and excitation (SE) (Hu, Shen, and Sun 2018) is inclusively searched. The version without SE is utilized to make comparisons within a shorter time. The other one is utilized to make fair comparisons with MnasNet (Tan et al. 2019) and EfficientNet (Tan and Le 2019).

Index	Expansion	Kernel Size	SE
0	3	3	-
1	3	3	✓
2	3	5	-
3	3	5	✓
4	3	7	-
5	3	7	✓
6	6	3	-
7	6	3	✓
8	6	5	-
9	6	5	✓
10	6	7	-
11	6	7	✓
12	-	1	-

Table 1: Each layer in our search space has 13 choices.

Scalable MBV2 Search Space. We utilize standard MobileNetV2 inverted bottleneck blocks (Sandler et al. 2018) to build up our search space after (Cai, Zhu, and Han 2019). We let the convolutional kernels be within $\{3, 5, 7\}$ and expansion rates $\{3, 6\}$, the channel of filters per layer is retained. On top of this, we include an identity block with linearly equivalent transformation for scalability. The size of this search space is 7^{19} .

Scalable MBV2 Search Space with SE. We align our search space by including Squeeze-and-excitation blocks with the latest works. Specifically, we give each MBV2 block an SE option, in total, we have 13 options per layer. The overall size of search space now becomes 13^{19} . The detailed choice blocks per layer are displayed in Table 1. Note that Index 12 refers to an identity block with the equivalent transformation (1×1 Conv). The rest choices are typical MBV2 blocks.

Pipeline

We utilize a multi-objective approach to serve as our pipeline (Chu, Zhang, and Xu 2019). We consider three objectives: classification error, multiply-adds and the number of parameters. We choose multiply-adds because we don't search models for specific hardwares. We also impose constraints for FLOPs to act as a mobile requirement. NSGA-II (Deb et al. 2002) is a powerful approach to address this kind of problem, so we start our work on top of it.

As (Zhang et al. 2018) states, mobile models are prone to underfitting instead of overfitting. Therefore, we try to maximize the number of parameters and utilize the weighted NSGA-II approach to weight different objectives as (Chu, Zhang, and Xu 2019). In a word, our problem can be defined as follows,

$$\begin{aligned}
 \min \quad & \{-\text{Accuracy}(m), \text{FLOPs}(m), -\text{Params}(m)\} \\
 \text{s.t.} \quad & m \in \text{search space } S \\
 & w_{acc} + w_{flops} + w_{params} = 1 \\
 & w_{acc}, w_{flops}, w_{params} \geq 0. \\
 & \text{FLOPs}(m) < 500M.
 \end{aligned} \tag{2}$$

The weight of the above three objectives are $w_{acc} = 0.4$, $w_{flops} = 0.4$, $w_{params} = 0.2$.

We use choice index to encode the chromosome. Therefore, we can use $m_1 = (x_1^1, x_2^1, \dots, x_{19}^1)$ to represent a model. Besides, we utilize the standard NSGA-II procedure, and only point out something different if necessary.

Initialization We initialize population to introduce various choice blocks to encourage exploration.

Crossover To be simple, we use single-point crossover.

Mutation We use a mixed approach: a PPO based controller to encourage exploitation (Schulman et al. 2017) and a Roulette wheel selection to encourage exploration. The hyper-parameter is listed in Table 2.

Weighted Non-dominated Sorting We weigh different preferences for various objectives by defining weighted crowding distance.

We run 120 epochs with population size of 70 to get 8400 models. This search stage takes about 1.5 GPU days on a Tesla V100. Then we sample 4 models from the final Pareto front with approximately equal crowding distance and train them completely.

Training Strategy

As for the training of the supernet, we use the same setting as FairNAS (Chu et al. 2019), except we train for 60 epochs and it takes about 10 GPU days.

As for the full training stage, we use the same configuration as FairNAS (Chu et al. 2019). And we use standard Inception pre-processing tricks (Szegedy et al. 2017). Unlike EfficientNet, we don't apply AutoAugment policy (Cubuk et al. 2018) because many state-of-the-art algorithms report their results without it. Moreover, we use RMSProp optimizer with 0.9 momentum. We use a batch size of 4096 and the initial learning rate of 0.256, which decays 0.01 every 2.4 epochs. We use a dropout rate of 0.2 (Srivastava et al. 2014) before the last FC layer and a weight decay ($l2$) rate of $1e-5$.

Table 2: Hyperparameters for the weighted NSGA-II approach.

ITEM	VALUE	ITEM	VALUE
POPULATION N	70	MUTATION RATIO	0.8
p_{rm}	0.2	p_{re}	0.65
p_{pr}	0.15	p_M	0.7
p_{K-M}	0.3		

Experiment Results

Comparison with State of the art methods

Table 3 gives a clear comparison of state-of-the-art models on ImageNet dataset. We pick models within the range

Methods	Mult-Adds (M)	Params (M)	Top-1 (%)	Top-5 (%)
MobileNetV2 1.0 (Sandler et al. 2018)	300	3.4	72.0	91.0
MobileNetV3 Large 1.0 (Howard et al. 2019)	219	5.4	75.2	92.2
MnasNet -A1 (Tan et al. 2019)	312	3.9	75.2	92.5
MnasNet-A2 (Tan et al. 2019)	340	4.8	75.6	92.7
FBNet-B (Wu et al. 2019)	295	4.5	74.1	
Proxyless-R Mobile (Cai, Zhu, and Han 2019)	320 [†]	4.0	74.6	92.2
Proxyless GPU (Cai, Zhu, and Han 2019)	465 [†]	7.1	75.1	-
Single-Path NAS (Stamoulis et al. 2019)	365	4.3	75.0	92.2
FairNAS-A (Chu et al. 2019)	388	4.6	75.3	92.4
MoGA-A (Chu, Zhang, and Xu 2019)	304	5.1	75.9	92.8
EfficientNet B0 (Tan and Le 2019)	390	5.3	76.3	93.2
SCARLET-A (Ours)	365	6.7	76.9	93.4
SCARLET-B (Ours)	329	6.5	76.3	93.0
SCARLET-C (Ours)	280	6.0	75.6	92.6

Table 3: Comparison of neural models on ImageNet. The input size is set to 224×224 . [†]: Based on its published code.

of FLOPs from 200M to 400M. It is clear that our SCARLET series marks a new state of the art, with SCARLET-A surpassing EfficientNet-B0 with +0.5% increase on top-1 accuracy and 25M fewer FLOPs, SCARLET-B achieves the same top-1 accuracy with 61M fewer FLOPs. Although both A and B have higher numbers of parameters, this treatment should be encouraged as it is related to representational power (Chu, Zhang, and Xu 2019) and doesn't necessarily increase inference latency. SCARLET-C also achieves a competing result with +0.3% increase on top-1 accuracy compared with FairNAS-A, while costing 108M fewer FLOPs.

SCARLET-A makes full use of large kernels (five 5×5 and seven 7×7 kernels) to enlarge receptive field. Besides it activates many squeezing and excitation (12 out of 19) blocks to improve its classification performance. At the early stage, it appreciates either large kernels and small expansion ratios or small kernels and large expansion ratios to balance the trade-off between accuracy and FLOPs.

SCARLET-B chooses two identity operations. Compared with A, it shortens network depth at the last stages. Besides, it utilizes squeezing and excitation block extensively (14 out of 17). It places a large expansion block with large kernels at the tail stage.

SCARLET-C uses three identity operations and utilizes small expansion ratio extensively to cut down the FLOPs, large expansion ratio at the tail stage whose resolution is 7×7 . It prefers large kernels before the downsampling layers. Besides, it makes an extensive use of squeeze and excitation to boost accuracy.

Ablation Study

Equivalent Transformation vs. Identity To check the validness of our method, we just utilize identity as a basic choice to act as the baseline group, which is commonly used in prior works. We train the two supernets under the same training setting for 60 epochs. We report the average running accuracy and standard variance based on the epoch scale in

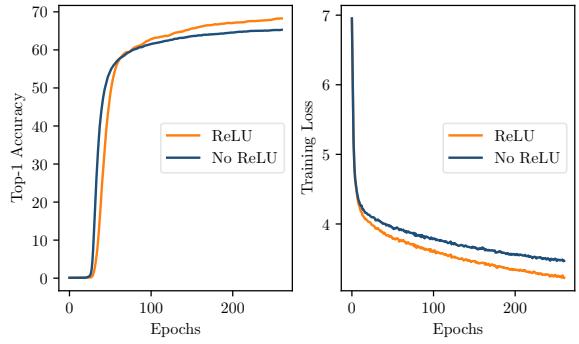


Figure 3: Training of a random model where identity blocks are enabled with linear transformation vs. with non-linear transformation.

Figure 1. Our method with linearly equivalent transformation can obtain about 20% higher than the baseline in case of the top-1 accuracy on the training set. Moreover, it has much lower variance, which indicates each model is trained fairly. To further verify this, we sample all the models in the last epoch (60-th) and report their metrics by a histogram, which is shown at the bottom of Figure 1. Identity makes troubles for the training and quite a few models suffer seriously and their metrics are below 30%. Therefore, they are severely under-estimated by the supernet. Whereas, LET can compensate and bring the models to a reasonable range.

Equivalent vs Non-equivalent Transformation Here we show that non-equivalent transformation changes the representative power of neural networks. A simple modification by adding ReLU function can violate the equivalence. To prove this, we randomly sample a model meta and then forcibly flip some choice blocks to identity : [1, 3, 1, 0, 12, 0, 0, 0, 12, 12, 12, 12, 12, 0, 0, 0, 12, 12, 9]. We train this model with ReLU (non-equivalent) and without (equiv-

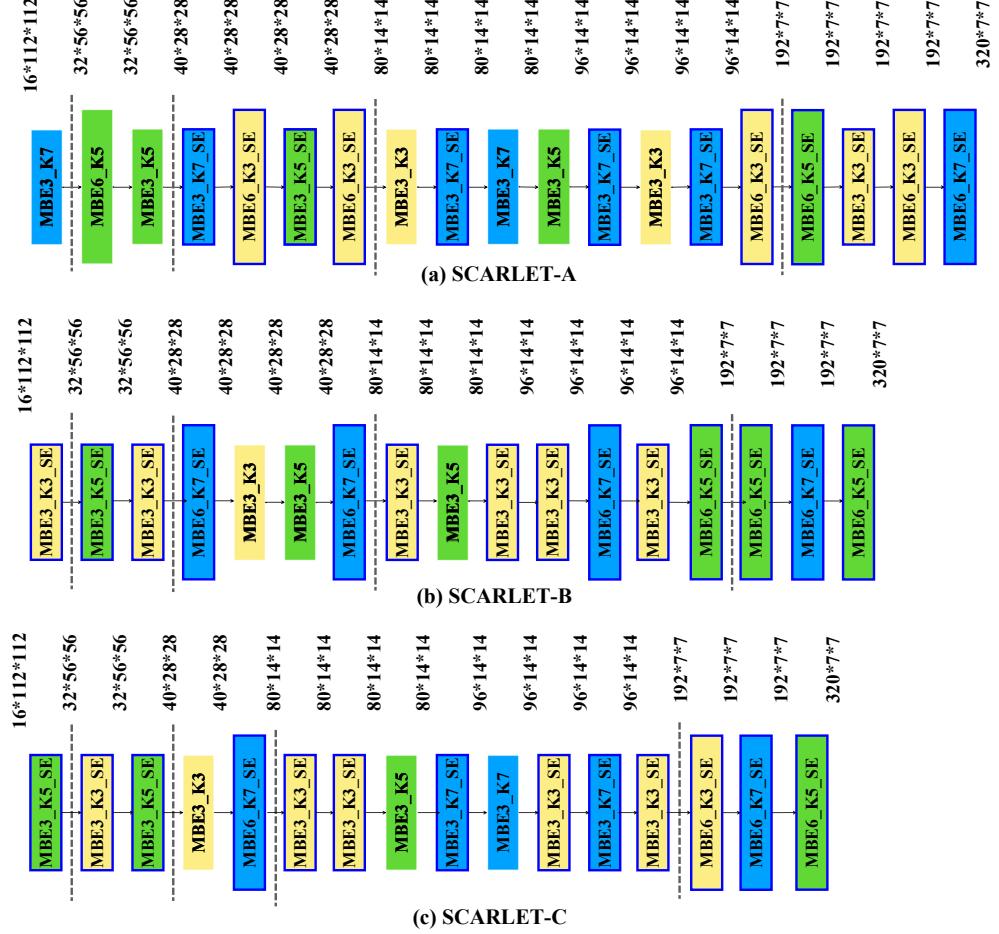


Figure 4: The architectures of SCARLET-A,B,C. Notice the dashed lines refer to downsampling points. The stem and tail parts are omitted. Best viewed in color.

alent) for identity layer³ on the basis of same seeds, training tricks, initialization strategy, and hyper-parameters. Figure 3 indicates that a trivial modification non-negligibly affects its representative power. While in our scalable supernet, we have to guarantee a equivalent transformation and this is why ReLU can't apply.

Discussion and Future Work

Weight sharing is one of the most critical features for efficient neural architecture search. Most of the one-shot approaches concentrate on how to find useful networks from choosing parallel choices. This schema hardly meets the requirement for flexibility, it even causes conflicts inherently. Since a neural network learns features layer by layer, it's highly sensitive to any scaling operation. Our equivalent transformation can be regarded as a buffer for such operations. Nevertheless, it works best under some conditions. In our search spaces, a linear transformation is used to match a single inverted bottleneck layer, where only two non-linear

activation functions are involved. When the matched function is too complicated, it will be more difficult to compensate.

How to perform flexible search efficiently remains open. Google's reinforced approach on top of huge computing resource (Zoph et al. 2018; Tan et al. 2019) is neither affordable nor environmental friendly. One of our future works is making the search process both flexible and efficient.

Conclusion

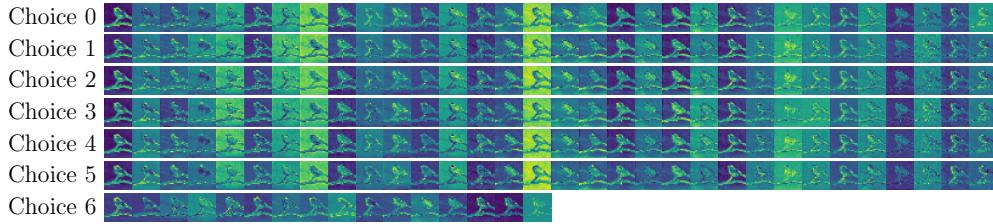
In this paper, we unveil the overlooked scalability issue in one-shot neural architecture search approaches. We show that simply adding identity blocks introduces training instability. By compensating the learning process with linearly equivalent transformation, we fill the gap between scalability and stability. We prove and demonstrate such transformation is identical in terms of representational power. The renewed supernet then can be trained with desired convergence and delivers competitive neural architectures. Namely, with fewer FLOPs than EfficientNet-B0,

³Note 12 means identity block.

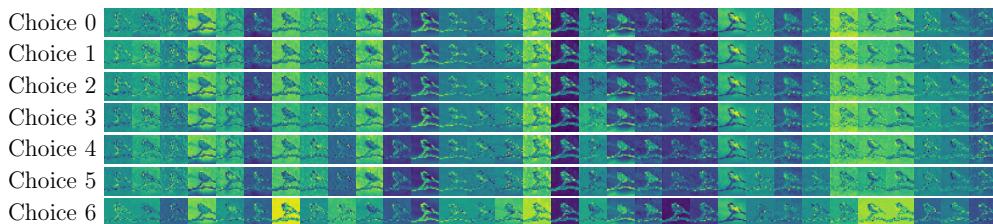
SCARLET-A achieves 76.9% Top-1 accuracy on ImageNet. SCARLET-B illustrates that shallow models can perform better which hits the same 76.3% as EfficientNet-B0 with much reduced FLOPs. SCARLET-C strikes 75.6%, also exceeds its peers of similar sizes.

References

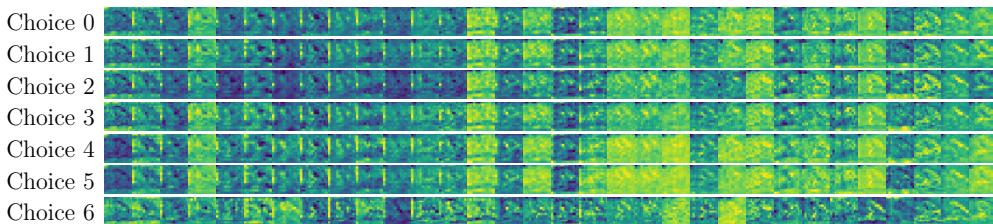
- [Bender et al. 2018] Bender, G.; Kindermans, P.-J.; Zoph, B.; Vasudevan, V.; and Le, Q. 2018. Understanding and simplifying one-shot architecture search. In *International Conference on Machine Learning*, 549–558.
- [Brock et al. 2018] Brock, A.; Lim, T.; Ritchie, J. M.; and Weston, N. 2018. Smash: one-shot model architecture search through hypernetworks. *International Conference on Learning Representations*.
- [Cai, Zhu, and Han 2019] Cai, H.; Zhu, L.; and Han, S. 2019. Proxylessnas: Direct neural architecture search on target task and hardware. In *International Conference on Learning Representations*.
- [Chen, Goodfellow, and Shlens 2015] Chen, T.; Goodfellow, I.; and Shlens, J. 2015. Net2net: Accelerating learning via knowledge transfer. *International Conference on Learning Representations*.
- [Chu et al. 2019] Chu, X.; Zhang, B.; Xu, R.; and Li, J. 2019. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. *arXiv preprint arXiv:1907.01845*.
- [Chu, Zhang, and Xu 2019] Chu, X.; Zhang, B.; and Xu, R. 2019. Moga: Searching beyond mobilenetv3. *arXiv preprint arXiv:1908.01314*.
- [Cubuk et al. 2018] Cubuk, E. D.; Zoph, B.; Mane, D.; Vasudevan, V.; and Le, Q. V. 2018. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*.
- [Deb et al. 2002] Deb, K.; Pratap, A.; Agarwal, S.; and Meiyarivan, T. 2002. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation* 6(2):182–197.
- [Deng et al. 2009] Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 248–255. Ieee.
- [Guo et al. 2019] Guo, Z.; Zhang, X.; Mu, H.; Heng, W.; Liu, Z.; Wei, Y.; and Sun, J. 2019. Single path one-shot neural architecture search with uniform sampling. *arXiv preprint arXiv:1904.00420*.
- [He et al. 2016] He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778.
- [Howard et al. 2019] Howard, A.; Sandler, M.; Chu, G.; Chen, L.-C.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; et al. 2019. Searching for mobilenetv3. *arXiv preprint arXiv:1905.02244*.
- [Hu, Shen, and Sun 2018] Hu, J.; Shen, L.; and Sun, G. 2018. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7132–7141.
- [Huang et al. 2018] Huang, Y.; Cheng, Y.; Chen, D.; Lee, H.; Ngiam, J.; Le, Q. V.; and Chen, Z. 2018. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *arXiv preprint arXiv:1811.06965*.
- [Sandler et al. 2018] Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; and Chen, L.-C. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4510–4520.
- [Schulman et al. 2017] Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.
- [Srivastava et al. 2014] Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.
- [Stamoulis et al. 2019] Stamoulis, D.; Ding, R.; Wang, D.; Lymberopoulos, D.; Priyantha, B.; Liu, J.; and Marculescu, D. 2019. Single-path nas: Designing hardware-efficient convnets in less than 4 hours. *arXiv preprint arXiv:1904.02877*.
- [Szegedy et al. 2017] Szegedy, C.; Ioffe, S.; Vanhoucke, V.; and Alemi, A. A. 2017. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*.
- [Tan and Le 2019] Tan, M., and Le, Q. V. 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. *International Conference on Machine Learning*.
- [Tan et al. 2019] Tan, M.; Chen, B.; Pang, R.; Vasudevan, V.; and Le, Q. V. 2019. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.
- [Wu et al. 2019] Wu, B.; Dai, X.; Zhang, P.; Wang, Y.; Sun, F.; Wu, Y.; Tian, Y.; Vajda, P.; Jia, Y.; and Keutzer, K. 2019. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- [Zagoruyko and Komodakis 2016] Zagoruyko, S., and Komodakis, N. 2016. Wide residual networks. *Proceedings of the British Machine Vision Conference*.
- [Zhang et al. 2018] Zhang, X.; Zhou, X.; Lin, M.; and Sun, J. 2018. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- [Zoph et al. 2018] Zoph, B.; Vasudevan, V.; Shlens, J.; and Le, Q. V. 2018. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8697–8710.



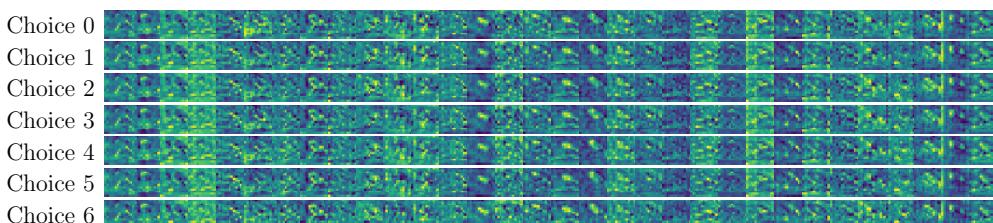
(a) Identity, first choice block



(b) Identity with LET, first choice block



(c) Identity, high-level choice block



(d) Identity with LET, high-level choice block

Figure 5: Learned low-level and high-level features with and without linearly equivalent transformation (LET).