Pattern Recognition 118 (2021) 108025

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Improving One-Shot NAS with Shrinking-and-Expanding Supernet☆

Yiming Hu[a,b], Xingang Wang[a], Lujun Li[a,b], Qingyi Gu[a,*]

[a] Institute of Automation, Chinese Academy of Sciences, East Zhongguancun Road, Haidian District, Beijing, China
[b] School of Artificial Intelligence, University of Chinese Academy of Sciences, Jingjia Road, Huairou District, Beijing, China

ARTICLE INFO

Article history:
Received 12 July 2020
Revised 29 April 2021
Accepted 1 May 2021
Available online 15 May 2021

Keywords:
Neural architecture search
Supernet
Search space shrinking

ABSTRACT

Training a supernet using a copy of shared weights has become a popular approach to speed up neural architecture search (NAS). However, it is difficult for supernet to accurately evaluate on a large-scale search space due to high weight coupling in weight-sharing setting. To address this, we present a shrinking-and-expanding supernet that decouples the shared parameters by reducing the degree of weight sharing, avoiding unstable and inaccurate performance estimation as in previous methods. Specifically, we propose a new shrinking strategy that progressively simplifies the original search space by discarding unpromising operators in a smart way. Based on this, we further present an expanding strategy by appropriately increasing parameters of the shrunk supernet. We provide comprehensive evidences showing that, in weight-sharing supernet, the proposed method SE-NAS brings more accurate and more stable performance estimation. Experimental results on ImageNet dataset indicate that SE-NAS achieves higher Top-1 accuracy than its counterparts under the same complexity constraint and search space. The ablation study is presented to further understand SE-NAS.

1. Introduction

Neural Architecture Search (NAS) is an important branch of automatic machine learning (AutoML), and has aroused growing interests due to its remarkable progress in various computer vision tasks [1–6]. It aims to reduce the cost of human efforts on manually designing network architectures and discover promising models automatically by balancing performance and resource constraint. Many early works of NAS are based on Reinforcement Learning (RL) [7,8], where a neural architecture is generated by agent's action and agent's reward is based on the evaluation performance of the trained architecture. Evolutionary algorithm (EA) is an alternative to RL for optimizing the neural architecture, that mutates well-performing models to explore better ones. However, these methods [9,10] need to sample and evaluate (from scratch) a large number of network architectures from the search space, which causes intensive computational overhead, e.g., hundreds of days with thousands of GPUs.

To speed up the searching process, ENAS [11] is the first to present weight-sharing mechanism for efficient NAS. Instead of training models from scratch, it defines the supernet, in which all models share one copy of weights. Another popular type of weight-sharing approaches are DARTS [12] and its variants [13–15], and they further parameterize the architecture distribution by a set of continuous variables, named architecture parameters. Such parameters and network weights are jointly optimized during supernet training by back-propagation. Finally, the optimal network architecture is chosen according to the optimized architecture parameters. In contrast, One-Shot NAS [16–18] belongs to a new paradigm, and it decouples the architecture search from supernet training. For training, instead of covering all weights of supernet, only a part of them are activated and optimized in one iteration. As less parameters are involved, it requires less memory and is more efficient. For searching, One-Shot NAS firstly samples many architectures by random search or evolutionary algorithms; then such candidates are evaluated by inheriting weights from the well-trained supernet; finally, some well-performing ones are chosen from the search space.

Though One-Shot NAS boosts searching efficiency significantly, it inherently suffers from terrible performance estimation in the evaluation procedure. Because it usually searches over a large and complicated network space covering billions of options for discovering the superior ones, in which supernet parameters are highly coupled. In this way, child models would interfere with each other

☆ This work is supported by the National Key R&D Program of China (2018YFD0400902, 2019YFB1311100), the National Natural Science Foundation of China, Grant No. 61673376 and the Scientific Instrument Developing Project of the Chinese Academy of Science, Grant No. YJKYYQ20200045.

* Corresponding author.
E-mail addresses: huyiming2016@ia.ac.cn (Y. Hu), xingang.wang@ia.ac.cn (X. Wang), lilujun2019@ia.ac.cn (L. Li), qingyi.gu@ia.ac.cn (Q. Gu).

https://doi.org/10.1016/j.patcog.2021.108025
0031-3203/

when sampled and trained in different iterations, and their accuracy in supernet is unavoidably averaged, so that boundaries between strong and weak architectures are blurred. As a result, it's difficult to get stable and accurate performance ranking of candidate models by a well-trained supernet [19]. PCNAS [20] identifies the same issue of One-Shot NAS and calls it Posterior Fading from a Bayesian perspective, i.e., the gap between true parameter posterior and proxy posterior increases with the number of models in supernet.

This paper aims to address the issue of weight coupling in One-Shot NAS. From a perspective on decreasing the degree of weight sharing, a shrinking-and-expanding supernet is presented. This is achieved via two strategies working at operation level and architecture level respectively. For the former, we propose a novel search space shrinking strategy by designing powerful evaluation criteria of candidate operators. Our approach progressively discards unpromising candidates from different layers and focuses on training those potentially-good ones, which reduces difficulties of supernet to accurately predict capability of child models. For the latter, some adjacent layer-wise search spaces of small size are firstly merged after shrinking. And then we appropriately increase supernet parameters by maintaining a copy of weights independently for each partial model in the merged search space, so that degree of weight sharing is further decreased. Actually, these two strategies share the same objective and complement each other. Based on them, the shared weights are deeply decoupled, and the performance ranking of candidate architectures become more stable and more accurate.

The effectiveness of our method is verified on three popular search spaces, which provides convincing empirical evidences. Experimental results on ImageNet dataset [21] show that SE-NAS surpasses existing state-of-the-art NAS methods with a clear margin and achieves higher Top-1 accuracy under the same complexity constraint and search space. Comprehensive experimental analysis and ablation study are also conducted to further understand characteristics of SE-NAS on NASBench-201 [22]. It is observed that the proposed shrinking-and-expanding supernet is able to provide more stable and more accurate performance estimation than its baseline. To sum up, the main contributions of this work are as follows:

- We propose a novel search space shrinking method by designing powerful evaluation criteria of candidate operators in weight-sharing setting.
- We provide a new perspective of alleviating weight coupling by appropriately expanding parameters of child models in supernet.
- Comprehensive results show that the proposed strategies of parameter decouple are effective, which improves the searching performance of one-shot NAS a lot.

2. Related Work

Manually designed neural networks have achieved significant success in a wide range of computer vision tasks [23–27]. However, it is widely believed that artificial architectures are sub-optimal. Recently, neural architecture search (NAS) has aroused increasing interests from both academia and industry.

2.1. Weight-Sharing NAS

To reduce computation, many methods train weight-sharing supernet for efficient NAS. They fall into two categories: One-Shot NAS [16–18,28] and gradient-based methods [12,29,30], which differ in the modeling process of network architectures.

Gradient-based algorithms introduce architecture parameters for each operator, and jointly optimize them and network weights by back-propagation. Finally, the optimal model is chosen from a well-trained supernet according to magnitudes of architecture parameters. Early gradient-based methods [12,13] suffer from two typical limitations. First, they need to utilize proxy tasks, such as training with images of low resolution, or learning with only a few blocks due to high consumption of GPU memory. ProxylessNAS [15] addresses the problem by keeping only two architectures in memory each time when training supernet. Second, the "winner takes all" effect causes premature convergence and reduces diversity of architectures. To this end, FairDARTS [31] offers each operator independent architecture weights to resolve their unfair competition.

One-Shot NAS belongs to a new paradigm. It firstly trains an over-parameterized supernet and then searches on the discrete search space covering lots of candidate models. During training stage, sample strategy matters since it decides how to train an effective supernet for performance estimation. For example, the early method [32] drops out operators with increasing probability when training supernet, so that weights of different models co-adapt. Based on this, SPOS [16] presents the uniform sampling strategy. For each iteration, only a single path is activated and gets optimized by regular gradient-based optimizers. FairNAS [28] strengthens SPOS by complying with strict fairness for both sampling and training of supernet. GreedyNAS [33] proposes a multi-path sampling strategy with rejection, and greedily discards those weak paths.

Different from previous NAS methods, our approach aims to decouple supernet parameters by reducing the degree of weight sharing. Besides, this work strikes a better trade-off between searching performance and cost compared to existing NAS methods.

2.2. Search Space Shrinking

Recent progressive approaches [20,34–36] explore the network space by gradually reducing the size of search space. They adopt progressive shrinking strategy for different purposes. To boost searching efficiency, PNAS [34] performs neural architecture search by gradually increasing network depth. Starting from the first cell, all possible block structures are trained and evaluated, in which well-performing ones are further expanded and unpromising ones are discarded. Afterwards, EPNAS [35] extends PNAS by presenting a new network transform policy with RE-INFORCE and an effective learning strategy. To reduce memory consumption, PDARTS [36] proposes search space approximation based on magnitudes of architecture parameters when progressively growing supernet depth. To improve ranking quality, PCNAS [20] drops out unpromising operators layer by layer according to supernet accuracy.

Though existing progressive methods have achieved decent results, it's still challenging to accurately detect unpromising operators among lots of candidate ones due to limited capability of supernet. Thus, how to design effective selection criterion of operators is crucial. This work presents a new strategy of assessing the potential of operators by taking operator importance, selection confidence and evaluation stability together into consideration, which is verified to benefit our shrinking method a lot. More importantly, this paper presents a complementary strategy for further alleviating weight coupling by properly expanding supernet parameters after shrinking step.

3. Proposed Method

In this section, One-Shot NAS [16,28] will be briefly revisited firstly, followed by our motivations. Then, parameters decouple strategies (i.e., search space shrinking and supernet parameters expanding) will be presented to improve One-Shot NAS.

Table 1
The mean Kendall's Tau of 10 repeat experiments on NASBench-201. Supernet accuracy is obtained by inferring a model in weight-sharing setting. Standalone accuracy comes from the model by normal training from scratch without weight sharing.

Metrics	CIFAR-10	CIFAR-100	ImageNet-16-120
supernet accuracy	0.543	0.532	0.539
standalone accuracy	0.853	0.842	0.845

3.1. Preliminary: One-Shot NAS

Our goal is to directly learn network architectures for large-scale target tasks without proxy task or dataset. One-Shot NAS is a good choice due to its low consumption of GPU memory. Taking MobileNet-like search space [15] as an example, the supernet $\mathcal{N}$ can be represented as a directed acyclic graph (DAG), in which each node represents feature map of a network layer. An edge $E_{i,j}$ can be regarded as information flow connecting node i and node j, which is composed of different operations such as MobileNetV2 blocks [37] with various kernel size and expand ratio. The operation space is defined as $\mathbb{O}$ and each edge has identical candidate operators, i.e., $|\mathbb{O}| = |\mathbb{O}_{i,j}|$. The whole search space can be represented as a discrete set $\mathbb{A}$.

In contrast to differentiable NAS, One-Shot NAS methods decouple supernet training from model search. In each iteration, a part of network weights sampled from $\mathcal{N}$ are updated for parameters co-adaptation. Supernet parameters can be optimized as:

$$\boldsymbol{W}^* = \arg\min_{\boldsymbol{W}} \mathcal{L}_{train}(\mathcal{N}(\mathbb{A}, \boldsymbol{W})), \tag{1}$$

where $\mathcal{L}_{train}$ is the loss function on the training dateset. The searching stage aims to find the network architecture that has the optimal performance on the validation set, as:

$$\begin{aligned} a^* &= \arg\max_{a\in\mathbb{A}} ACC_{val}(\mathcal{N}(a, \boldsymbol{W}^*(a))), \\ s.t.&\quad Cost(a^*) < C, \end{aligned} \tag{2}$$

where $ACC_{val}(\cdot)$ denotes supernet accuracy on the validation set. The architecture a inherits parameters from the well-trained supernet weights $\boldsymbol{W}^*$ as $\boldsymbol{W}^*(a)$. $Cost(a^*)$ represents additional objectives (e.g., latency or FLOPs constraint) on hardware.

3.2. Motivation

One-Shot NAS boosts searching efficiency, while it suffers from two limitations.

First, the performance estimation of supernet is inaccurate. To evaluate this, we leverage Kendall's ranking correlation coefficient, i.e., Kendall's Tau (τ), which defines a measure of correlation between two ranks. Specifically, the supernet constructed on NASBench-201 [22] is firstly trained following [16]. Then all child models are ranked according to supernet accuracy and the ground truth provided by NASBench-201 respectively. Finally, the Kendall's Tau between such two types of ranks is computed as ranking correlation of supernet accuracy. The ranking correlation of standalone accuracy (i.e., normal training from scratch without weight sharing) is also computed for comparison. Table 1 shows comparison results on three different datasets. Supernet accuracy is over 0.3 lower than standalone accuracy in ranking correlation on all three datasets, which suggests that supernet is far from a satisfying performance indicator.

Second, the model ranking based on supernet suffers limited stability. To verify this, we conduct 10 repeated experiments and get ranks of randomly selected 30 child models from NASBench-201 according to supernet accuracy and standalone accuracy respectively. For each child model, we retrieve its ranking values in different runs, and visual their ranking distributions as in Fig. 1. Here, X-axis means child models are ordered with ground truths provided by NASBench-201 and Y-axis represents model ranks based on metrics. It can be found that model ranking based on standalone accuracy is highly correlated to its ground-truth ranking. In contrast, supernet accuracy has a wider ranking range in multiple runs. This suggests weight-sharing supernet is unstable as a performance estimator, which makes One-Shot NAS difficult to reproduce.

Assuming inaccuracy and instability of supernet are modeled as a function ϕ and an unbiased noise q (i.e., prediction randomness) respectively, the issues of One-Shot NAS can be simply formulated as:

$$\begin{aligned} ACC_{val}(\mathcal{N}(a, \boldsymbol{W}^*(a))) &= \phi(ACC_{val}(\boldsymbol{w}_a)) + q, \\ \mathbb{E}(q) &= 0, \end{aligned} \tag{3}$$

where $\boldsymbol{w}_a$ denotes the weights obtained by sampling and training the standalone model a. Obviously, existence of ϕ and q will hurt performance prediction of supernet. Theoretically, appropriately reducing the degree of weight sharing is a fundamental method to relieve negative effects of ϕ and q. Motivated by this, we present SE-NAS by building a shrinking-and-expanding supernet for decoupling the shared parameters, which performs search space shrinking (see Section 3.3) at operation level and expanding supernet parameters (see Section 3.4) at architecture level respectively.

3.3. Search Space Shrinking

Search space shrinking is a feasible method to mitigate weight coupling, which can reduce redundancy of supernet and improve its training efficiency at the same time. However, it is computationally intractable to enumerate all network architectures and evaluate their performance in a large-scale search space. Hence shrinking search space at architecture level is impractical. Instead, pruning operators seems to be a feasible solution. Assuming a given operator is considered to be weak, it can be dropped out from the search space. And those child models containing the operator are also discarded together. One of the most important concerns of search space shrinking at operation level is to judge capability of operators by an accurate metric. PDARTS [36] adopts shrinking methods at operation level to discard poor operators according to the magnitude of architecture parameters. One-Shot NAS methods have no architecture parameters like PDARTS, so evaluation criteria of candidate operators need to be defined at first.

Candidate operators evaluation. Before describing the shrinking pipeline, alternative operators in the search space is firstly evaluated. As One-Shot NAS methods based on search space shrinking have to detect unpromising operators by performance ranks of child models on the validation set, so how to evaluate operators based on supernet matters. The shrinking strategy aims to improve the ability of supernet to rank models, but it still suffers terrible performance ranking in early shrinking stage due to serious weight coupling. In order to prevent important operators from being removed from supernet, we present a new evaluation criterion of candidate operators by considering evaluation stability, selection importance and selection confidence at the same time:

- Evaluation Stability. One trivial way of alleviating negative effects of q is to train multiple supernets and then ignore the noise by computing their expectation. However, supernet training and evaluation require extra computation cost. Thus we approximate multiple samples by supernet parameters at different iterations of the last epoch. Then the averaged parameters are used to evaluate the child model a:

$$P(a) = ACC_{val}(\frac{1}{K}\sum_{k}^{K} \boldsymbol{W}_k^*(a)), \tag{4}$$

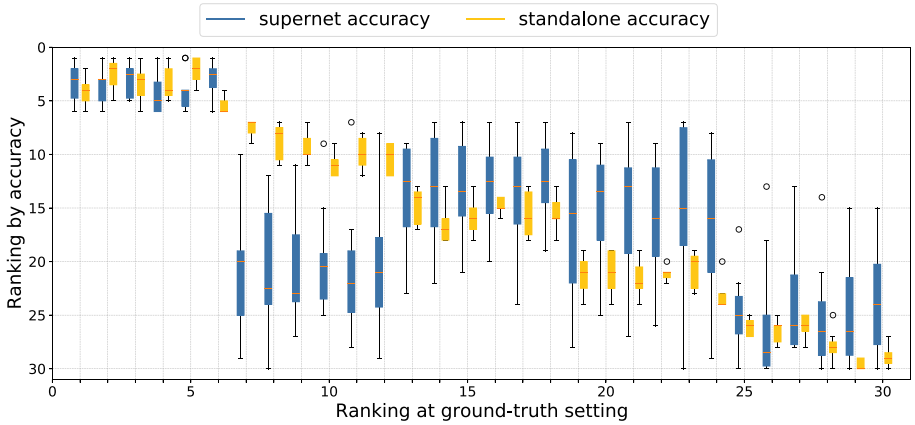

Fig. 1. Ranking distribution for each child model with CIFAR-10 on NASBench-201. X-axis represents ground-truth ranks sorted from the best to the worst. Each box extends from the lower to upper quartile values of model ranking according to different metrics (standalone accuracy and supernet accuracy). The median is marked with a line. The whiskers show the ranking range. Outliers are marked with circles.

where $\boldsymbol{W}_k^*(a)$ represents the weight sampled at certain mini-batch of the last epoch.

- Selection Importance. The selection importance of an operator o is further defined as the mean accuracy of child models containing the operator:

$$SI_o = \mathbb{E}_{a\in\{a|a\in\mathbb{A},a \text{ contains } o\}}(P(a)), \tag{5}$$

where $o \in \mathbb{O}_{i,j}$, a is uniformly sampled from the sample set $\{a|a \in \mathbb{A}, a \text{ contains } o\}$. In practice, instead of computing the mean accuracy precisely across the whole search space, we sample finite number of child models containing the operator randomly, and then compute their mean accuracy instead. In Equation (5), the operators with low importance are more likely to come from some weak architectures, and such operators would encourage worse models in the search space, so they can be discarded safely. From this point of view, the evaluation criterion is quite reasonable, since it adequately considers the combination of an operator with other operators from different layers by computing mean accuracy of multiple child models.

- Selection Confidence. As in Section 3.2, empirical $ACC_{val}(\boldsymbol{W}^*(a))$ causes model averaging effects and bias the performance evaluation of standalone models. In particular, weight sharing would narrow performance gap of different models in the search space. So ranking operators based on SI may cause undesired inconsistency with ground-truth ranking due to limited capability of supernet accuracy. To eliminate the impacts of ϕ on search space shrinking, we further introduce the layer variance, which reflects selection confidence of each layer and is leveraged to reduce uncertainty of operator selection. Assuming there's large performance gap among different candidates in a layer, the operators with low importance in this layer can be discarded safely. That is to say, priority is always given to those layers with higher confidence (i.e., bigger variance) when pruning operators. Concretely, the confidence of an operator is defined as variance of candidates in corresponding layers:

$$SC_o = \mathrm{Var}(SI_o), \quad o \in \mathbb{O}_{i,j}. \tag{6}$$

In this work, the score of an operator is defined according to the above three factors together. Intuitively, those operators with low selection importance SI and high selection confidence SC would be dropped out from the search space. Therefore, the final score of an operator o is defined as follows:

$$S_o = \frac{normalize(SI_o)}{normalize(SC_o)}, \tag{7}$$

where normalize() represents mean normalization. In our approach, S_o is leveraged to rank operators and guide the whole shrinking process. Compared with existing shrinking methods [20] that simply adopt supernet accuracy as a metric, the proposed evaluation criteria are able to help supernet discover unpromising operators precisely.

Removing unpromising operators As shown in Fig. 2, we divide the shrinking pipeline into multiple stages and progressively discard unpromising candidate operators. All operators (points) on the graph are alive before the first shrinking step. The initial training of supernet follows [16] with uniform sampling strategy. After this, all operators in the operation pool are evaluated according to Equation (7). Then those operators with the lowest K scores are removed from $\mathbb{A}$ and corresponding highlights in Fig. 2 are turned off, which means they will not participate in the later training. The training of shrunk supernet is identical to the initial training procedure. The only distinction lies in that the shrunk supernet builds layer-wise search spaces of different size and flexibly samples operators for each layer. In this way, only well-performing candidate operators survive and those redundant ones in the original search space are removed. The number of child models is gradually decreasing, as if the optimal model distribution moves to a small area, which can improve the ranking ability of supernet due to the reduction of weight coupling. It's worthy noticing that at least one operator is preserved for each edge during the shrinking process, since SE-NAS should not change the topology of supernet. When the process of search space shrinking finishes, the shrunk supernet can be further applied to architecture search. Finally, the optimal network architecture is retrained from scratch. The above pipeline of search space shrinking can be easily generalized to other cell-based search spaces.

3.4. Supernet Parameters Expanding

Layer-wise search space gradually shrinks as redundant operators are removed by steps, enabling weight-sharing supernet to be decoupled. Based on this, the strategy of expanding supernet parameters aims at further decoupling the shared parameters, so that the performance gap between a standalone model and the model in weight sharing setting is bridged. Simply copying independent weights for certain layers of a child model can decouple it from other models partially, which pushes its weights more close to weights of the model in standalone setting. Ideally, such two types of weights will be equivalent if all layers of the model have their own weights instead of inheriting weights from supernet. In

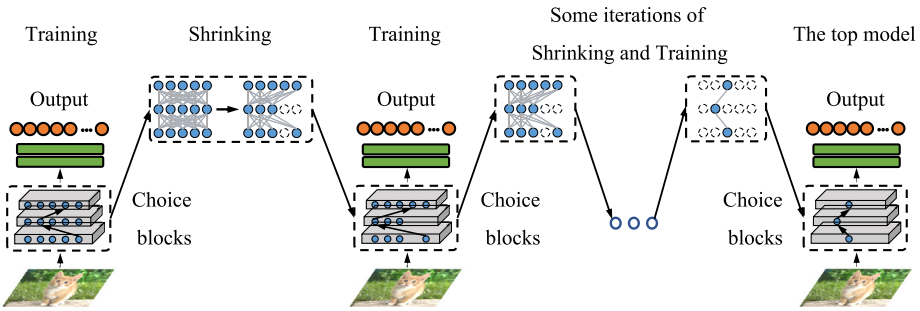

Fig. 2. Pipeline of the proposed method with search space shrinking. Supernet is firstly trained for some epochs with uniform sampling. After this, all operators are ranked by their scores according to Equation (7). Some operators are removed from the search space by steps if their rankings fall at the tail.

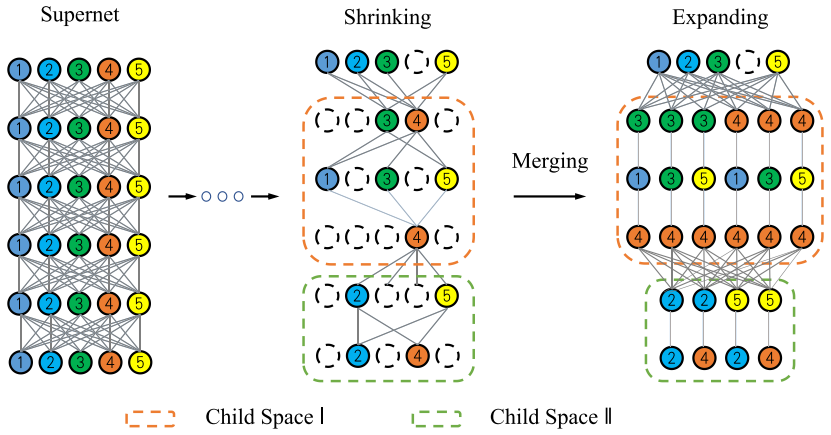

Fig. 3. A toy example of expanding supernet parameters. The algorithm starts with search space shrinking, followed by search space merging and supernet parameters expanding. Here, the threshold of search space merging is set to 6 and two merged child spaces are generated.

view of this, expanding supernet parameters shares the same objective (i.e., reduce the degree of sharing weight and decouple the shared weights) as the strategy of search space shrinking, and they complement each other.

Specifically, in the process of search space shrinking, we merge some layer-wise search spaces of small size and continue to deeply decouple the shared weights in the merged search spaces. Adjacent layer-wise search spaces are merged from shallow to deep layers gradually when the size of the child search space to be merged is no more than a threshold. Supernet may contain multiple such child search spaces without overlaps among them, which will be generated successively. The key is that each partial model in the merged search space maintains one copy of weights and will be trained independently. A toy example of expanding supernet parameters is shown in Fig. 3. The number ranging from 1 to 5 represents types of candidate operators for each layer. And the initial supernet contains 6 layers and each layer is assigned 5 operators. The strategy of search space shrinking reduces the size of layer-wise network spaces gradually. After several shrinking steps, the number of candidate operators for some layers begins to be less than 5. Here, the threshold of search space merging is set to 6, so 2 child search spaces are generated by merging adjacent layer-wise search space, i.e., from the second to the fourth layer and from the fifth to the sixth layer. In the merged child space 1 or child space 2, each partial model does not share any parameters with other models. The child model belonging to the merged search space has only two or three layers, thus is called partial model.

The training strategy remains unchanged except that the decoupled supernet maintains more copy of weights for the merged child search spaces. Supernet parameters are gradually expanded along with the shrinking process, which makes supernet accuracy of a model more close to its true performance in standalone setting. At last, supernet will become almost perfect performance estimator without sharing weight, when the size of the whole search space begins to be less than the predefined threshold along with shrinking steps. The threshold can be simply set to 50, which could be flexibly adjusted according to computation resources. It has to be emphasized that expanding supernet parameters doesn't increase GPU memory consumption, because only one architecture is used and stored in memory at each iteration due to the uniform sampling strategy. The whole pipeline of SE-NAS is illustrated in Algorithm 1 .

Algorithm 1: SE-NAS

Input: A supernet $\mathcal{N}$, the threshold of merging layer-wise search spaces $\mathcal{T}$, the number of operators dropped out per iteration k.
Output: the optimal network architecture from $\mathcal{N}$

```
while (stopping conditions are not satisfied) do
    Training the supernet N with uniform sampling strategy following [17];
    Computing scores of operators from N according to Equation (7);
    Removing k operators from N with the lowest k scores;
    Merging adjacent layer-wise search spaces successively, and generating a set of merged child spaces {A_i}, where size of A_i is not more than T;
    for A_i in {A_i} do
        Expanding supernet parameters by maintaining a copy of weights independently for each partial model in A_i.
    end
end
```

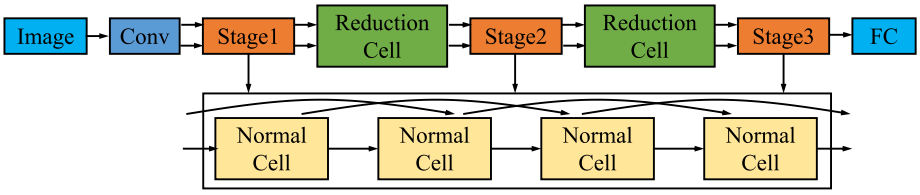

Fig. 4. The skeleton of DARTS search space.

Table 2
The skeleton of MobileNet-like search space.

Input Shape	Block Type	Output Channel	Repeat	Stride
$224^2 \times 3$	3×3 conv	40	1	2
$112^2 \times 40$	CBS	24	1	1
$112^2 \times 24$	CBS	32	4	2
$56^2 \times 32$	CBS	56	4	2
$28^2 \times 56$	CBS	112	4	2
$14^2 \times 112$	CBS	128	4	1
$14^2 \times 128$	CBS	256	4	2
$7^2 \times 256$	CBS	432	1	1
$7^2 \times 432$	1×1 conv	1728	1	1
$7^2 \times 1728$	GP	-	1	1
$1^2 \times 1728$	FC	1000	1	-

3.5. Search Space

Instead of searching a proxy network architecture with fewer building blocks, this paper conducts neural architecture search on the target network space. Next, three different search spaces adopted in our method will be introduced, including NASBench-201 [22], DARTS-based and MobileNet-like search spaces [12,15].

MobileNet-like Search Space. The MobileNet-like search space used in this paper is identical to ProxylessNAS [15] and its skeleton is shown in Table 2. The candidate block structure (CBS) adopts Inverted Bottleneck of MobileNetV2 [37]. Each layer contains 6 CBS with different kernel size and expansion ratio, plus one identity operation. In addition, the expansion factor in the first CBS block is set to 1, and the identity operation is not used in the first block of every stage. The size of search space is 7^{21}.

DARTS Search Space. The skeleton of DARTS search space starts with one 3×3 convolution layer. The main body of the skeleton is divided into three stages, connected by several reduction cells with a stride of 2. Normal cell is stacked 4 times in each stage. The skeleton ends up with a classification layer to transform network outputs into the final prediction. Here, a cell has an ordered sequence of 6 nodes (feature maps) and each node connects all previous nodes. Every edge connecting two nodes is associated with 8 operations (3×3 and 5×5 separable convolutions, 3×3 and 5×5 dilated convolutions, 3×3 max and average pooling, identity and none). Each cell has two input nodes and a single output node. The input nodes are defined as the outputs of previous two cells. The output of a cell is obtained by concatenating all the intermediate nodes. The search space is shown in Fig. 4 and its size is 8^{14}.

NASBench-201 Search Space. The search space shares the similar skeleton as DARTS. It differs from DARTS in details. For example, NASBench-201 stacks 5 normal cells in each stage. It replaces the reduction cell by a residual block with a stride of 2. In addition, each cell is composed of 4 nodes and the operation set only incorporates 5 candidates (1×1 and 3×3 convolutions, 3×3 average pooling, identity and none), which is shown in Fig. 5. The search space trains 15625 models from scratch and provides their ground-truth performance, which allows researchers to focus on the search algorithm itself without unnecessary repetitive training for the searched model.

4. Experiments

In this section, the power of SE-NAS is demonstrated from three aspects by experiments: first, adequate experiments are conducted to verify and analyze the effectiveness of the proposed shrinking-and-expanding supernet in ranking correlation with ground truth and ranking stability. All analysis experiments are conducted on NASBench-201 [22], since it provides the real performance of all architectures, which can be treated as the ground truth labels; second, it is shown that the searched models by SE-NAS are able to consistently outperform existing NAS algorithms (SPOS [16], FairNAS [28], FBNet [13], ProxylessNAS [15], PCNAS [20], DARTS [12]) with a large margin on MobileNet-like and DARTS-based search spaces respectively; third, detailed ablation studies further demonstrate the impacts of each component in our method.

4.1. Dataset

ImageNet. In the paper, all comparison experiments are performed on ILSVRC2012, CIFAR-10 and CIFAR-100 datasets. ImageNet [21] is a large-scale dataset which contains over 1.2 million training images and 50,000 validation images belonging to 1000 classes. Following previous works [16], the original training set is randomly splitted into two parts: 50000 images for validation and the rest as training set. In our experiments, supernet is trained on training set and operator scores are computed on validation set. The original validation set is used as a test set to measure the final performance of the searched network architecture.

CIFAR. CIFAR-10 dataset contains 50,000 training images and 10,000 test images spanning 10 categories, while CIFAR-100 dataset has the identical number of images spanning 100 categories. For these two datasets, 5000 images are randomly selected from the original training set as validation set and the others as training set.

4.2. Analysis

Ability of SE-NAS to reserve promising architectures. To verify the effectiveness of the proposed shrinking-and-expanding supernet, three different metrics (random, accuracy-based and magnitude-based metrics) are compared on NASBench-201. In our settings, the ground-truth score of each operator is computed by averaging ground-truth accuracy of all models containing the given operator, and the ground-truth ranking is obtained based on the ground-truth scores. Alternative operators are ranked according to corresponding metric-based scores, in which the operator score in our approach is defined as Equation (7) ; for the magnitude-based metric, the magnitude of architecture parameters are directly used as scores to rank operators following DARTS; the baseline with random scores of operators is also set for comparison. Our approach divides the process of supernet parameters decouple into two stages, where each stage discards 10 operators with the lowest scores respectively.

As shown in Fig. 6, SE-NAS retains most of operators with the top 21 ground-truth rankings in the first shrinking stage. For the

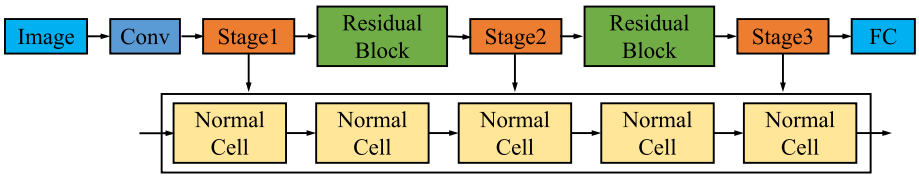

Fig. 5. The skeleton of NASBench-201 search space.

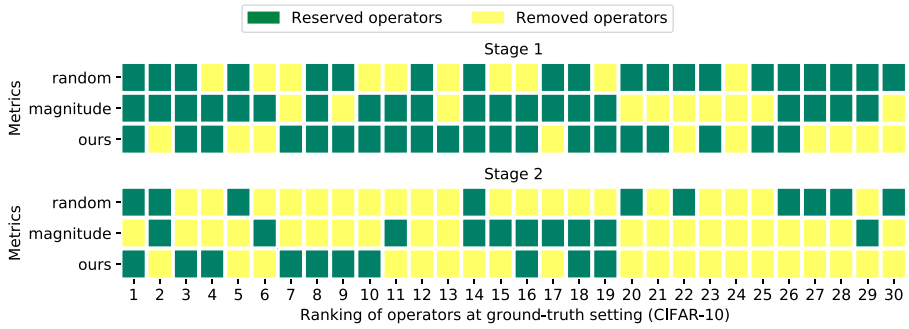

Fig. 6. The operator distribution at different stages of supernet parameters decouple on NASBench-201 with CIFAR-10. The decouple process is divided into two stages and each stage discards 10 operators.

second stage, both random and magnitude-based metrics remove most of operators in the top 10 ground-truth rankings, while SE-NAS reserves most of them. Moreover, our approach makes sure that most of reserved operators in the second shrinking stage have higher ground-truth scores than the removed ones. Actually, there's no guarantee that child models with the optimal performance must be hidden in the shrunk search space that contains the operators with top ground-truth rankings. While it's reasonable to believe SE-NAS is more likely to discover well-behaved architectures from the shrunk search space. Because the operators with the top scores tend to come from some well-performing architectures, and such operators would encourage superior network architectures in the search space, so SE-NAS preserves them.

Ranking correlation of SE-NAS. Due to the weight coupling problem of One-Shot NAS, there's weak correlation between supernet accuracy and the ground-truth accuracy. So in this section, the ability of SE-NAS to rank models is verified by comparing ranking correlation of the decoupled supernet with its baseline [16]. To quantify ranking correlation of a supernet, we rank child models in the current search space according to supernet accuracy, and then compute the Kendall ranking correlation coefficient (Kendall's Tau for short) between the ranking and the ground-truth ranking provided by NASBench-201. In detail, our approach divides the process of supernet parameters decouple into two stages and each stage trains supernet for 20 epochs. For some epochs, child models in the shrunk supernet are ranked according to the accuracy in weight-sharing setting and the ground-truth accuracy respectively. Then the Kendall's Tau between these two types of ranks is computed as the ranking correlation coefficient. Without shrinking and expanding supernet parameters, the baseline computes the ranking correlation coefficient at some epochs based on its own supernet.

Fig. 7 shows comparison results of ranking correlation on three different datasets (CIFAR-10, CIFAR-100, ImageNet-16-120). It can be found that SE-NAS gradually achieves higher ranking correlation than its baseline at the end of the first stage of parameters decouple. After the second parameters decouple stage, ranking correlation of SE-NAS surpasses the baseline with a clear margin on all three datasets, which suggests that the shrinking-and-expanding supernet provides more accurate performance estimation than the original supernet when evaluating the capability of child models.

Ranking stability of SE-NAS. It has been shown that SE-NAS is able to improve ranking correlation of supernet. In this section, ranking stability of the shrinking-and-expanding supernet is further discussed. To verify this, 10 independent repeat experiments are conducted on three different datasets (CIFAR-10, CIFAR-100, ImageNet-16-120) respectively. Then the distributions of ranking correlation of SE-NAS and the baseline [16] in different runs are visualized. In detail, the process of supernet parameters decouple is divided into two stages and each stage trains supernet for 20 epochs. After two parameters decouple stages, the ranking correlation of the child models in the shrunk search space is computed based on the decoupled supernet. And the baseline uses its own supernet to compute the ranking correlation coefficient.

As Fig. 8 shows, the ranking correlation of our approach is more stable than the baseline in different runs. Obviously, it has much smaller variance and higher mean than its baseline on all three datasets, which demonstrates SE-NAS has more stable model ranking by reducing candidate operators of search space and expanding supernet parameters. Additionally, the stabilized evaluation criterion (Equation (4)) also contributes a lot to the current results. In practice, stability is a crucial advantage for NAS and can relieve the problem of reproducibility for weight-sharing NAS methods.

4.3. Overall Comparison Results

In this section, comprehensive experiments are conducted to show the power of SE-NAS by comparing with different NAS methods (e.g., FBNet [13], ProxylessNAS [15], DARTS [12], FairNAS [28], SPOS [16]) on three popular search spaces.

NASBench-201. SE-NAS is compared with various NAS approaches on NASBench-201. In detail, the score of each operator is computed by Equation (7). The process of parameters decouple is divided into two stages and supernet is trained for 20 epochs in each stage. SE-NAS searches on validation and test set of CIFAR-10 only and then transfers the searched models to other datasets.

From Table 3, it can be observed that our approach achieves consistent performance gain on all three datasets. SE-NAS outperforms all NAS approaches except GDAS with a clear margin, while it achieves comparable results with GDAS using about one third of time. Besides, our approach has almost reached the upper limit (optimal) of performance on NASBench-201.

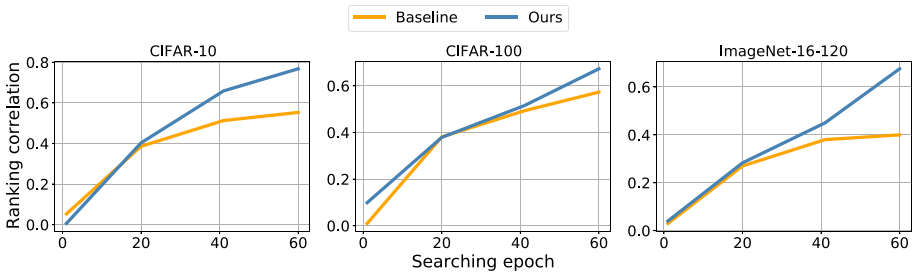

Fig. 7. The evolution of ranking correlation on NASBench-201. Baseline refers to SPOS [16] without parameter decouple. Our method decouples supernet parameters at 20-th epoch and 40-th epoch respectively.

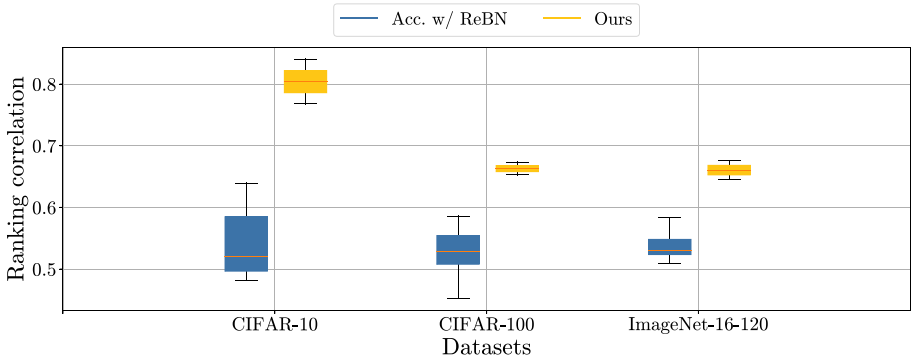

Fig. 8. The distribution of ranking correlation of 10 repeated experiments on NASBench-201. Baseline refers to SPOS [16]. The Y-axis represents the range of ranking correlation coefficient from different repeated experiments.

Table 3
Searching results on NASBench-201. 6 different searching algorithms are compared on the search space. The mean accuracy of searched models is reported for different NAS methods. Training cost is computed by running each algorithm on a single GPU (GeForce GTX 1080 Ti) with a batch size of 256.

Method	Time (seconds)	CIFAR-10 Acc. (%)		CIFAR-100 Acc. (%)		ImageNet-16-120 Acc. (%)	
		validation	test	validation	test	validation	test
RSPS [38]	8007.1	80.42 ± 3.58	84.07 ± 3.61	52.12 ± 5.55	52.31 ± 5.77	27.22 ± 3.24	26.28 ± 3.09
DARTS-V1 [12]	11625.7	39.77 ± 0.00	54.30 ± 0.00	15.03 ± 0.00	15.61 ± 0.00	16.43 ± 0.00	16.32 ± 0.00
DARTS-V2 [12]	35781.8	39.77 ± 0.00	54.30 ± 0.00	15.03 ± 0.00	15.61 ± 0.00	16.43 ± 0.00	16.32 ± 0.00
GDAS [39]	31609.8	89.89 ± 0.08	93.61 ± 0.09	71.34 ± 0.04	70.70 ± 0.30	41.59 ± 1.33	41.71 ± 0.98
SETN [18]	34139.5	84.04 ± 0.28	87.64 ± 0.00	58.86 ± 0.06	59.05 ± 0.24	33.06 ± 0.02	32.52 ± 0.21
ENAS [11]	14058.8	37.51 ± 3.19	53.89 ± 0.58	13.37 ± 2.35	13.96 ± 2.33	15.06 ± 1.95	14.84 ± 2.10
Optimal	N/A	91.61	94.37	73.49	73.51	46.77	47.31
Ours	10561.1	90.50 ± 0.09	93.47 ± 0.14	71.14 ± 0.10	71.91 ± 0.19	44.96 ± 0.94	45.66 ± 1.05

MobileNet-like search space. The process of supernet parameters decouple is divided into multiple stages. Supernet is trained for 100 and 5 epochs in the first and other stages. The same training setting as [42] is adopted for training supernet. Specifically, we use a stochastic gradient descent (SGD) optimizer with momentum 0.9 and weight decay 4×10^{-5}. The batch size is 1024 and the initial learning rate is 0.5. The learning rate of each iteration is equal to the initial learning rate multiplying $1 - \frac{e}{T}$, where e and T represent the current iteration and the number of total iterations respectively. The score of each operator is computed based on Equation (7), in which 1000 models containing the given operator are randomly sampled for approximating sample average of all candidates. When sampling child models, SE-NAS dumps models that don't satisfy FLOPs constraint. For FairNAS, SE-NAS removes one operator for each layer each time because of its fair constraint. For other NAS algorithms, SE-NAS removes 10 operators whose rankings fall at the tail in each shrinking cycle. For each searched model under different efficiency constrains, the Top-1 accuracy and standard deviation of three independent repeated experiments are reported. In the re-training phrase, the training settings and hyper-parameters of all searched models follow supernet training. All searched network architectures are trained from scratch for 240 epochs (300000 iterations).

(1) Searching results under the constraint of FLOPs. As shown in Table 4, it can be observed that training cost of SE-NAS is much less than that of its counterparts on ImageNet, which indicates SE-NAS brings significant efficiency improvement by shrinking search space, since it progressively reduces the size of search space. More importantly, the proposed method achieves better searching performance than its baselines under different FLOPs constraints. For example, SE-NAS discovers the network architectures from the MobileNet-like search space with at least 0.3%, 0.2%, 0.9% and 0.6% higher accuracy than other NAS methods at the FLOPs level of 250M, 320M, 470M and 590M respectively. Noticing that the search space and setting of SE-NAS-D follow PCNAS [20], which adds 3 more kinds of operators on basis of MobileNet-like search space. Meanwhile, the detailed architectures of searched networks are also visualization for verification in Fig. 9, 11, 12. Some interesting phenomena can be observed: first, SE-NAS prefers to place the bottleneck blocks with large kernels and expand ratio nearby

Table 4
ImageNet results on DARTS and MobileNet-like search space under constraint of FLOPs. The average accuracy and standard deviation of searched models are reported. The structures of models searched by SE-NAS are visualized in Fig. 9, 11, 12. ⋆ searched on CIFAR10. †: reported in GreedyNAS [33].

Architecture	Search Method	Shrinking	Search Space	FLOPs (M)	Params. (M)	Top-1 Acc. (%)	Time (GPU-days)
MobileNetV2-1.0× [37]	manual	×	-	300	-	72.0	-
FBNet-A [13]	gradient	×	MobileNet	249	4.9	73.0	9
SE-NAS-A	gradient	✓	MobileNet	250 ± 1	4.7 ± 0.1	**73.30 ± 0.07**	7
ProxylessNAS-R [15]	gradient	×	MobileNet	320	3.8	74.6	15†
FairNAS [28]	EA	×	MobileNet	321	4.4	74.7	12†
SE-NAS-B	gradient	✓	MobileNet	321 ± 1	3.9 ± 0.2	**74.86 ± 0.10**	7
Random	-	×	MobileNet	472 ± 6	5.2 ± 0.9	72.77 ± 1.09	-
SPOS [16]	EA	×	MobileNet	466	7.1	74.8	12 †
ProxylessNAS [15]	gradient	×	MobileNet	465	7.1	75.1	15 †
SE-NAS-C	gradient	✓	MobileNet	470 ± 2	6.0 ± 0.5	**76.01 ± 0.13**	7
ShuffleNetV2-2.0× [26]	manual	×	-	591	-	74.9	-
NASNet-A [40]	RL	×	NASNet	564	5.3	74.0	1800
PCNAS [20]	gradient	✓	MobileNet	595	5.1	76.1	-
DARTS [12]	gradient	×	DARTS	595	4.9	73.1	4⋆†
GDAS [39]	gradient	×	DARTS	581	5.3	74.0	0.2
SNAS [41]	gradient	×	DARTS	522	4.3	72.7	1.5
SE-NAS-D	gradient	✓	MobileNet	592 ± 2	5.4 ± 0.2	**76.75 ± 0.09**	7
SE-NAS-E	gradient	✓	DARTS	597 ± 3	5.2 ± 0.4	**74.72 ± 0.15**	8

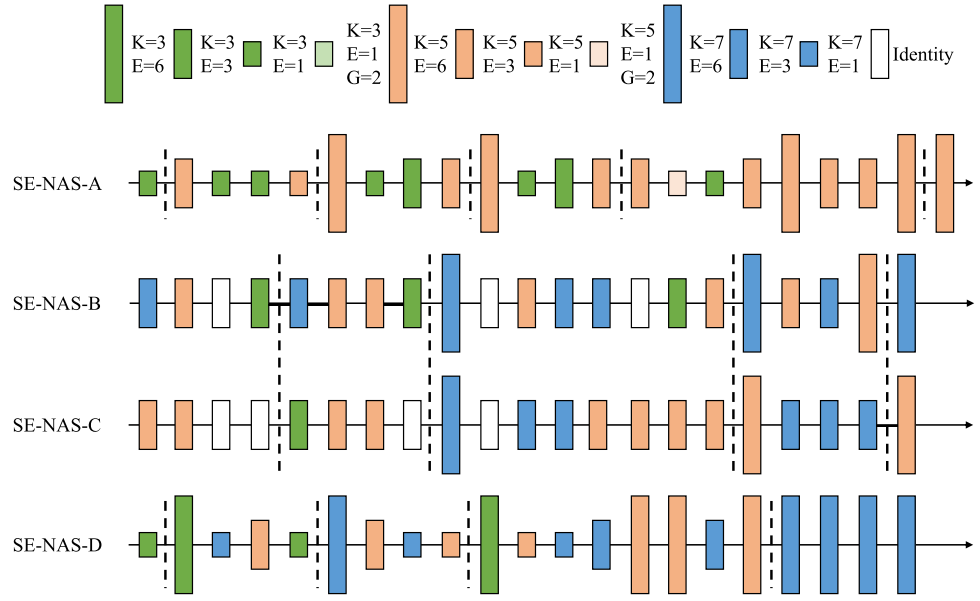

Fig. 9. visualization of the searched architectures under the constraint of FLOPs. Rectangle boxes are used to represent blocks for each layer. The kernel size of the depth-wise convolution is represented with different colors, blue for kernel size of 7, orange for kernel size of 5, green for kernel size of 3, light color for group size of 2 and empty for "Identity" operator. The height of rectangle boxes represents the expansion rate of the block. Totted line denotes down-sample layers with stride of 2.

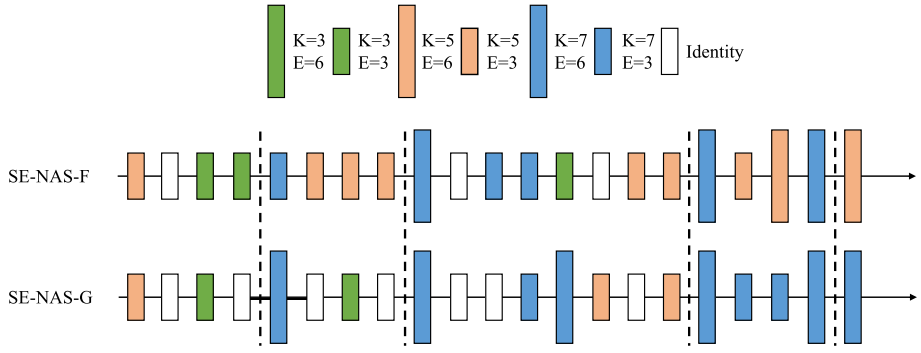

Fig. 10. visualization of the searched architectures under the constraint of latency. Rectangle boxes are used to represent blocks for each layer. The kernel size of the depth-wise convolution is denoted with different colors, blue for kernel size of 7, orange for kernel size of 5, green for kernel size of 3, and empty for "Identity" operator. The height of rectangle boxes represents the expansion rate of the block. Totted line denotes down-sample layers with stride of 2.

Table 5
ImageNet results on MobileNet-like search space under the constraint of latency. The average Top-1 accuracy and standard deviation of searched models are reported. The searched network architectures are visualized in Fig. 10. †: reported in GreedyNAS [33].

Architecture	Search Method	Shrinking	Latency (ms)	Params. (M)	Top-1 Acc. (%)	Time (GPU-days)
ProxylessNAS-R [15]	gradient	×	17	3.8	74.6	15†
SPOS [16]	EA	×	17	4.1	74.8	12†
SE-NAS-F	gradient	✓	16 ± 1	3.9 ± 0.3	**74.79 ± 0.08**	7
SPOS [16]	EA	×	22	7.1	75.3	12 †
ProxylessNAS [15]	gradient	×	22	7.1	75.1	15 †
SE-NAS-G	gradient	✓	23 ± 0	6.3 ± 0.2	**75.87 ± 0.10**	7

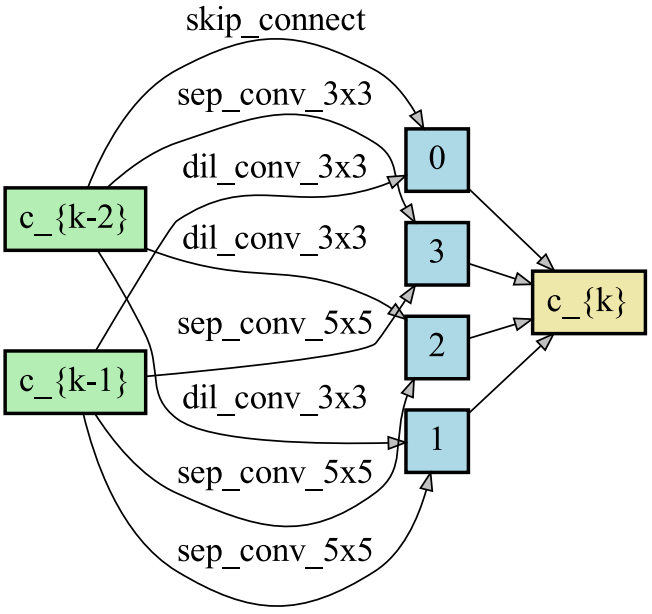

Fig. 11. Normal cell of SE-NAS-E.

the down-sampling block or later stages to preserve more information. Second, the models searched by SE-NAS tend to be deeper, which differentiates our approach from existing NAS methods. Actually, the result is reasonable: the deeper model can extract more powerful features, since functions that can be represented compactly by deep architectures cannot be represented by a compact shallow architecture [43]. Besides, the searching results of random search are also reported. It is worthy of noticing that SE-NAS outperforms random search with a clear margin, which demonstrates the strong ability of our method in discovering powerful network architectures.

(2) Searching results under the constraint of latency. Searching experiments are further conducted to compare searching performance of SE-NAS and others under the constraint of latency. The results are summarized in Table 5. In details, the target latency is set to 17ms and 22ms respectively according to our measurement of mobile setting models on GPU. Our searched model on the MobileNet-like search space, namely SE-NAS-G, achieves 75.87% Top-1 accuracy under the constraint of 22ms latency, which is over 0.7% higher than its counterpart. Compared with SPOS, SE-NAS-F achieves comparable Top-1 accuracy with less training cost under the constraint of 17ms latency. The network architectures of searched models are visualized in Fig. 10. It can be found that the searched models show different preferences under different searching constraints: the models searched under the constraint of latency tend to skip more layers for saving the inference time. For example, SE-NAS-C has 17 layers while SE-NAS-G only has 14 layers.

DARTS search space. The effectiveness of SE-NAS over DARTS search space is demonstrated in this part. In details, we apply the search procedure on CIFAR-10, and then retrain the searched models from scratch on ImageNet. Supernet is trained for 200 and 5 epochs in the first and other stages respectively. The initial training of supernet adopts much more training epochs due to slow convergence of supernet in DARTS search space. Similarly, SE-NAS removes 10 operators from different edges each time and builds edge-wise search spaces. The re-training stage follows the training setting of [36]. As reported in Table 4, SE-NAS brings significant improvement (1.6% Top-1 accuracy) compared with its baseline on DARTS search space. The network architectures of searched model are visualized in Fig. 11 and Fig. 12. It can be observed that the architecture discovered by DARTS has lots of non-parametric operators (e.g., Identity and Pooling), while SE-NAS prefers to replace them with more parametric operators, which allows the searched models to be deeper.

4.4. Ablation Study

In this section, the impacts of different components of SE-NAS are further studied. Here, SPOS [16] is set as the baseline, and it trains supernet constructed on the MobileNet-like search space with uniform sampling strategy. To make clear the contribution of each component, searching results of the baseline are compared by combining with different policies. The experimental setting follows Section 4.3.

The overall comparison results are reported in Table 6. It can be found that searched network architecture of SPOS equipped with the strategy of search space shrinking achieves 75.3% Top-1 accuracy on ImageNet, which is 0.5% higher than its baseline with comparable FLOPs. To investigate the impact of layer variance (Equation (6)), the searching experiment is further conducted

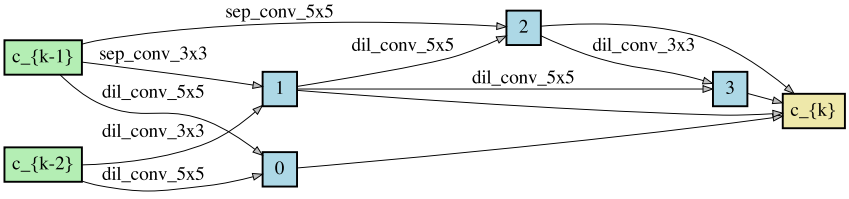

Fig. 12. Reduction cell of SE-NAS-E.

Table 6
Ablation Study on ImageNet. SI refers to operator importance (see Equation (5)). SC denotes selection confidence (see Equation (6)).

Methods	Params.	FLOPs	Top-1 Acc.	Top-5 Acc.
SPOS [16]	7.1M	466M	74.8%	91.5%
SPOS+Shrinking (SI)	6.6M	468M	75.3%	92.3%
SPOS+Shrinking (SI + SC)	6.6M	467M	75.7%	92.5%
SPOS+ Shrinking (SI + SC) + Expanding	6.5M	469M	**76.1%**	**93.0%**

by considering SI and SC at the same time during evaluation, which brings 0.4% Top-1 accuracy improvement. Equipped with all components, Top-1 accuracy of the searched network architecture comes to 76.1%, which surpasses all other setting with a clear margin.

5. Conclusion and Future Work

In this paper, we point out the problem of existing One-Shot NAS algorithms. Due to weight coupling, weight-sharing supernet leads to unstable and inaccurate performance estimation. To address this problem, a shrinking-and-expanding supernet is presented for decoupling supernet parameters. We make a profound study on characteristics of the decoupled supernet by conducting comprehensive analysis experiments on NASBench-201. The power of SE-NAS is demonstrated by comparing with existing NAS algorithms on various search spaces and datasets. All experimental results prove that the proposed method helps to alleviate the dark side of One-Shot NAS, thus can improve its searching performance significantly.

It is believed that NAS will be a popular direction of automatic machine learning (AutoML) for a long time. As a partner technique, hyper-parameter optimization (HPO) is also an important topic in AutoML, with the aim of discovering appropriate hyper-parameters to enhance the performance of model training. Previous works usually apply such two techniques separately or directly search different network architectures with the same hyper-parameters. However, such optimization will lead to sub-optimal results, since both architecture and hyper-parameter matter and different architectures prefer their own training hyper-parameters. Thus, we think the joint optimization of architecture and hyper-parameter is a promising direction in the future. Inevitably, the joint search space is combinatorially large and challenging, making it hard to efficiently discover the optimal combination of architecture and hyper-parameter. To improve searching efficiency, candidate models can be efficiently evaluated under a proxy with computationally reduced setting instead of training them from scratch, e.g., FBNetV3 [44] uses neural acquisition function (i.e., predictor) to predict architecture statistics. AutoHAS [45] trains a weight sharing supermodel across architecture and hyper-parameter. Besides, we think improving sample efficiency using some powerful black box optimization methods [46] is also a possible direction for accelerating searching process in large joint search space.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] M. Zhang, Y. Zhou, J. Zhao, S. Xia, J. Wang, Z. Huang, Semi-supervised block-wisely architecture search for efficient lightweight generative adversarial network, Pattern Recognition 112 (2021) 107794.
[2] J. Liu, S. Zhou, Y. Wu, K. Chen, W. Ouyang, D. Xu, Block proposal neural architecture search, IEEE Transactions on Image Processing 30 (2020) 15–25.
[3] Y. Chen, Z. Wang, Z.J. Wang, X. Kang, Automated design of neural network architectures with reinforcement learning for detection of global manipulations, IEEE Journal of Selected Topics in Signal Processing 14 (5) (2020) 997–1011.
[4] Z. Ding, Y. Chen, N. Li, D. Zhao, Z. Sun, C.P. Chen, Bnas: Efficient neural architecture search using broad scalable architecture, IEEE Transactions on Neural Networks and Learning Systems (2021).
[5] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A.L. Yuille, L. Fei-Fei, Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 82–92.
[6] H. Jiang, F. Shen, F. Gao, W. Han, Learning efficient, explainable and discriminative representations for pulmonary nodules classification, Pattern Recognition 113 (2021) 107825.
[7] B. Zoph, Q.V. Le, Neural architecture search with reinforcement learning, in: International Conference on Learning Representations, 2016.
[8] Y. Chen, G. Meng, Q. Zhang, S. Xiang, C. Huang, L. Mu, X. Wang, Renas: Reinforced evolutionary neural architecture search, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 4787–4796.
[9] Q. Ye, Y. Sun, J. Zhang, J.C. Lv, A distributed framework for ea-based nas, IEEE Transactions on Parallel and Distributed Systems (2020).
[10] Z. Zhong, Z. Yang, B. Deng, J. Yan, W. Wu, J. Shao, C.-L. Liu, Blockqnn: Efficient block-wise neural network architecture generation, IEEE transactions on pattern analysis and machine intelligence (2020).
[11] H. Pham, M. Guan, B. Zoph, Q. Le, J. Dean, Efficient neural architecture search via parameters sharing, in: International Conference on Machine Learning, 2018, pp. 4095–4104.
[12] H. Liu, K. Simonyan, Y. Yang, Darts: Differentiable architecture search, in: International Conference on Learning Representations, 2019.
[13] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, K. Keutzer, Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 10734–10742.
[14] Y. Xu, L. Xie, W. Dai, X. Zhang, X. Chen, G.-J. Qi, H. Xiong, Q. Tian, Partially–connected neural architecture search for reduced computational redundancy, IEEE Transactions on Pattern Analysis and Machine Intelligence (2021).
[15] H. Cai, L. Zhu, S. Han, Proxylessnas: Direct neural architecture search on target task and hardware, in: International Conference on Learning Representations, 2019.
[16] Z. Guo, X. Zhang, H. Mu, W. Heng, Z. Liu, Y. Wei, J. Sun, Single path one-shot neural architecture search with uniform sampling, in: Proceedings of the European Conference on Computer Vision, 2020, pp. 818–833.
[17] M. Zhang, H. Li, S. Pan, X. Chang, C. Zhou, Z. Ge, S.W. Su, One-shot neural architecture search: Maximising diversity to overcome catastrophic forgetting, IEEE Transactions on Pattern Analysis and Machine Intelligence (2020).
[18] X. Dong, Y. Yang, One-shot neural architecture search via self-evaluated template network, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 3681–3690.
[19] Y. Zhang, Z. Lin, J. Jiang, Q. Zhang, Y. Wang, H. Xue, C. Zhang, Y. Yang, Deeper insights into weight sharing in neural architecture search, arXiv preprint arXiv:2001.01431 (2020).
[20] X. Li, C. Lin, C. Li, M. Sun, W. Wu, J. Yan, W. Ouyang, Improving one-shot nas by suppressing the posterior fading, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2020, pp. 13836–13845.
[21] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Communications of The ACM 60 (6) (2017) 84–90.
[22] X. Dong, Y. Yang, Nas-bench-201: Extending the scope of reproducible neural architecture search, in: International Conference on Learning Representations, 2020.
[23] T.-B. Xu, P. Yang, X.-Y. Zhang, C.-L. Liu, Lightweightnet: Toward fast and lightweight convolutional neural networks via architecture distillation, Pattern Recognition 88 (2019) 272–284.
[24] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, T. Chen, Recent advances in convolutional neural networks, Pattern Recognition 77 (2018) 354–377.
[25] P.V. Arun, I. Herrmann, K.M. Budhiraju, A. Karnieli, Convolutional network architectures for super-resolution/sub-pixel mapping of drone-derived images, Pattern Recognition 88 (2019) 431–446.
[26] N. Ma, X. Zhang, H.-T. Zheng, J. Sun, Shufflenet v2: Practical guidelines for efficient cnn architecture design, in: Proceedings of the European Conference on Computer Vision, 2018, pp. 116–131.
[27] M. Yousef, K.F. Hussain, U.S. Mohammed, Accurate, data-efficient, unconstrained text recognition with convolutional neural networks, Pattern Recognition 108 (2020) 107482.

[28] X. Chu, B. Zhang, R. Xu, J. Li, Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search, arXiv preprint arXiv:1907.01845 (2019).
[29] J. Chang, Y. Guo, M. Gaofeng, Z. Lin, S. XIANG, C. Pan, et al., Data: Differentiable architecture approximation with distribution guided sampling, IEEE Transactions on Pattern Analysis and Machine Intelligence (2020).
[30] N. Wang, S. XIANG, C. Pan, et al., You only search once: Single shot neural architecture search via direct sparse optimization, IEEE Transactions on Pattern Analysis and Machine Intelligence (2020).
[31] X. Chu, T. Zhou, B. Zhang, J. Li, Fair darts: Eliminating unfair advantages in differentiable architecture search, in: Proceedings of the European Conference on Computer Vision, Springer, 2020, pp. 465–480.
[32] G. Bender, P.-J. Kindermans, B. Zoph, V. Vasudevan, Q. Le, Understanding and simplifying one-shot architecture search, in: International Conference on Machine Learning, 2018, pp. 549–558.
[33] S. You, T. Huang, M. Yang, F. Wang, C. Qian, C. Zhang, Greedynas: Towards fast one-shot nas with greedy supernet, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2020, pp. 1999–2008.
[34] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, K. Murphy, Progressive neural architecture search, in: Proceedings of the European Conference on Computer Vision, 2018, pp. 19–34.
[35] J.-M. Perez-Rua, M. Baccouche, S. Pateux, Efficient progressive neural architecture search, in: British Machine Vision Conference, 2018, p. 150.
[36] X. Chen, L. Xie, J. Wu, Q. Tian, Progressive differentiable architecture search: Bridging the depth gap between search and evaluation, in: Proceedings of International Conference on Computer Vision, 2019, pp. 1294–1303.
[37] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, Mobilenetv2: Inverted residuals and linear bottlenecks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4510–4520.
[38] L. Li, A. Talwalkar, Random search and reproducibility for neural architecture search., in: UAI, 2019, p. 129.
[39] X. Dong, Y. Yang, Searching for a robust neural architecture in four gpu hours, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 1761–1770.
[40] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, Q.V. Le, Mnasnet: Platform-aware neural architecture search for mobile, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 2820–2828.
[41] S. Xie, H. Zheng, C. Liu, L. Lin, Snas: stochastic neural architecture search, in: International Conference on Learning Representations, 2018.
[42] Y. Hu, Y. Liang, Z. Guo, R. Wan, X. Zhang, Y. Wei, Q. Gu, J. Sun, Angle-based search space shrinking for neural architecture search, in: Proceedings of the European Conference on Computer Vision, Springer, 2020, pp. 119–134.
[43] Y. Bengio, Y. LeCun, et al., Scaling learning algorithms towards ai, Large-scale kernel machines 34 (5) (2007) 1–41.
[44] X. Dai, A. Wan, P. Zhang, B. Wu, Z. He, Z. Wei, K. Chen, Y. Tian, M. Yu, P. Vajda, J.E. Gonzalez, Fbnetv3: Joint architecture-recipe search using neural acquisition function, arXiv preprint arXiv:2006.02049 (2020).
[45] X. Dong, M. Tan, A.W. Yu, D. Peng, B. Gabrys, Q.V. Le, Autohas: Efficient hyperparameter and architecture search, arXiv preprint arXiv:2006.03656 (2020).
[46] L. Wang, S. Xie, T. Li, R. Fonseca, Y. Tian, Sample-efficient neural architecture search by learning actions for monte carlo tree search., IEEE Transactions on Pattern Analysis and Machine Intelligence (2021).

Yiming Hu received the B.Sc. degree from China University of Geosciences, Wuhan, China, in 2016. He is currently pursuing the Ph.D. degree in control science and engineering with the Research Center of Precision Sensing and Control, Institute of Automation, Chinese Academy of Sciences, Beijing, China, and the University of Chinese Academy of Sciences, Beijing. His current research interests include computer vision and neural architecture search.

Xingang Wang received the B.Sc. degree from Tianjin University, Tianjin, China, in 1995, and the Ph.D. degree from the Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, Changchun, China, in 2002. He is currently a Professor with the Research Center of Precision Sensing and Control, Institute of Automation, Chinese Academy of Sciences, Beijing, China. His current research interests include image processing and machine learning.

Lujun Li received the B.Sc. degree from Central South University, Changsha, China, in 2019. He is currently pursuing the Ph.D. degree in control science and engineering with the Research Center of Precision Sensing and Control, Institute of Automation, Chinese Academy of Sciences, Beijing, China, and the University of Chinese Academy of Sciences, Beijing. His current research interests include computer vision and machine learning.

Qingyi Gu (M'13) received the B.E. degree in electronic and information engineering from Xi'an Jiaotong University, Xi'an, China, in 2005, and the M.E. and Ph.D. degrees in engineering from Hiroshima University, Hiroshima, Japan, in 2010 and 2013, respectively. He is currently a Professor with the Institute of Automation, Chinese Academy of Sciences, Beijing, China, and the University of Chinese Academy of Sciences, Beijing. His primary research interests include high-speed image processing and applications in industry and biomedicine.