Efficient Re-parameterization Operations Search for Easy-to-Deploy Network Based on Directional Evolutionary Strategy

Yu Xinyi
  Zhejiang University of Technology

Wang Xiaowei
  Zhejiang University of Technology

Rong Jintao
  Zhejiang University of Technology

Zhang Mingyang
  Zhejiang University of Technology

Ou Linlin (✉ linlinou@zjut.edu.cn)
  Zhejiang University of Technology

Research Article

Efficient Re-parameterization Operations Search for Easy-to-Deploy Network Based on Directional Evolutionary Strategy

Xinyi Yu[1], Xiaowei Wang[1], Jintao Rong[1], Mingyang Zhang[1] and Linlin Ou[1*]

[1]College of Information and Engineering, Zhejiang University of Technology, Hangzhou, 310014, China.

*Corresponding author(s). E-mail(s): linlinou@zjut.edu.cn;
Contributing authors: 2112003049@zjut.edu.cn;

Abstract

Structural re-parameterization (Rep) methods has achieved significant performance improvement on traditional convolutional network. Most current Rep methods rely on prior knowledge to select the re-parameterization operations. However, the performance of architecture is limited by the type of operations and prior knowledge. To break this restriction, in this work, an improved re-parameterization search space is designed, which including more type of re-parameterization operations. Concretely, the performance of convolutional networks can be further improved by the search space. To effectively explore this search space, an automatic re-parameterization enhancement strategy is designed based on neural architecture search (NAS), which can search a excellent re-parameterization architecture. Besides, we visualize the output features of the architecture to analyze the reasons for the formation of the re-parameterization architecture. On public datasets, we achieve better results. Under the same training conditions as ResNet, we improve the accuracy of ResNet-50 by 1.82% on ImageNet-1k.

Keywords: neural network, evolutionary algorithm, structural re-parameterization, deep learning

1 Introduction

Neural architecture search (NAS)[1–8] has been widely used in many field, such as object detection[9, 10], semantic segmentation[11, 12], image recognition[13], image generation[14] and object re-recognition[15, 16]. Although the architecture searched by NAS is superior to traditional networks in performance such as VGG[17], ResNet[18], MobileNet[19–21] and DenseNet[22], its landing ability is far inferior to these networks, which makes it difficult for NAS methods to be widely-used in practice.

NAS tends to retain multi-path structures and more "shortcut" operations without artificial restrictions. This structure is not friendly to most terminal devices. Meanwhile, the architecture searched based on the DARTS[1] search space achieves better performance on the benchmark dataset, but some convolution operations (such as 7×7 and 5×5 separable convolutions, 7x1–1x7 sequential convolution) in the search space are not well optimized on the terminal devices, which makes them[2–4, 6, 23–25] not well deployed on edge devices. In Fig. 1, we tested the accuracy and inference time of networks on the NVIDIA embedded board. The networks searched by traditional search space do not perform well when deployed to edge devices. Especially, the accuracy and inference speed are inferior to the manually designed networks on ImageNet-1K. Therefore, it is a challenging task to improve the performance of the architecture without sacrificing the ability of deployment to terminal devices.

The structural re-parameterization technology provides us with a new idea. Rep methods effectively improve the performance of the tranditional networks, but the specific operations need to be selected based on prior knowledge[26, 27], such as depth, the type of convolution operations and the number of convolution operations. This approach makes the re-parameterization network suboptimal. To further improve the performance of the model and find
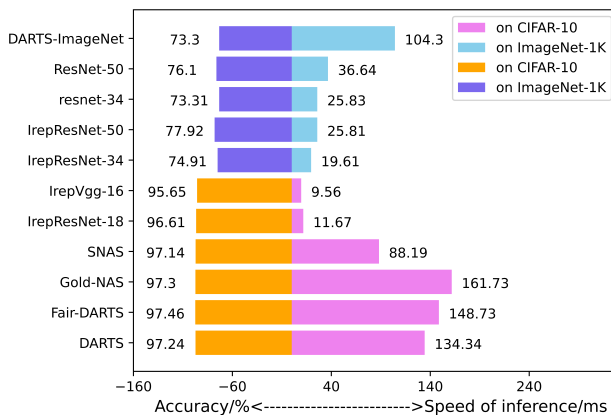

Fig. 1 The inference speed and accuracy of networks are tested on NVIDIA AgX Xavier. On ImageNet-1K, our networks achieved better accuracy and inference speed on the general dataset

the globally optimal networks, we use NAS method to search a set of re-parameterization operations, which can be fused into the traditional networks. In this way, we can get great performance and landing friendly models. In addition, the performance of re-parameterization models is closely related to the type of re-parameterization operations. Some previous work[28, 29] search the best combination of re-parameterization operations automatically, but the type of re-parameterization operations limits the upper limit of network performance. Therefore, the performance of model can be further improved by exploring more re-parameterization operations.

In addition to the re-parameterization operations involved in the above work, there are other convolution operations that can be fused into VGG-style networks after appropriate transformation. Thus, an improved re-parameterization search space (IREPS) was designed to make it include more re-parameterization operations, and a better set of re-parameterization operations can be searched from the larger search space, which can be fused into easily deployable networks without accuracy degradation.

For the larger re-parameterization search space, a directional evolutionary strategy (DES) was designed to explore a optimal architecture population from it. When training the SuperNet parameters, it learns the importance of blocks and candidate operations, and uses them as an indicator to generate different offspring architectures. Thus, the search strategy can directly ignore bad architectures, which makes the algorithm converge rapidly. To explain the reasons for the formation of the architecture and the improvement of performance, finally, we visualized the architecture and the output feature. In summary, the contributions can be summarized as follows:

1. An improved re-parameterization search space was designed, which contains more re-parameterization operations. Compared with other Rep search space, it can further improve the performance of traditional convolutional network.
2. To explore the larger search space, DES is proposed to search a set of re-parameterization operations. This search strategy makes a trade-off between the diversity of architectures and the efficiency of search.
3. Extensive experiments on image classification and its downstream tasks demonstrate our architecture achieve better results compared to other related work.

2 Related Work

2.1 Network Architecture Search

Neural architecture search (NAS) is a widely-used technique, which aims to search feature extraction networks that match given tasks. Evolutionary algorithm-based NAS[5, 7, 8, 30, 31] uses the principle of "survival of the fittest" to select architectures and rely on genetics, mutation, crossover and random generation to obtain new offsprings. Evolutionary algorithm is

well-established global optimization method with high robustness and wide applicability. However, evolutionary algorithm-based NAS converges slowly due to the random generation of offspring architectures.

Gradient-based NAS[1, 2, 4, 6, 32–36] benefits from the introduction of differentiable function, which transforms the discrete search space into continuous, so that it can be optimized by gradient optimization algorithm. From the perspective of parameter optimization, it can be divided into two categories. One is bilevel optimization[1, 6, 33, 35, 36], which optimizes the architecture parameters under the weight parameters are optimal. It can be described as:

$$\min_{\alpha} \mathcal{L}_{val}\left(w^*(\alpha), \alpha\right) \ \textbf{s.t.} \ w^*(\alpha) = \operatorname{argmin}_w \mathcal{L}_{\text{train}}\left(w, \alpha\right) \qquad (1)$$

where α denotes architecture, w_{α} denotes the network weight bound with the architecture α, L_{train} and L_{val} denote optimization loss on training and validation dataset. It first optimizes the network weights w, then finds α that minimizes the validation loss $\mathcal{L}_{val}$. The other is single-level optimization[2, 34, 37], which regards the optimization of w and α as independent processes. It can be described as:

$$\alpha^t, w^t + = \eta \nabla_{\alpha, w} \mathcal{L}_{\text{train}}\left(\alpha^{t-1}, w^{t-1}\right) \qquad (2)$$

Eq. 2 indicates that both w and α are optimized in an optimization process. Although gradient-based NAS can converge quickly, the existence of Matthew's effect makes architecture lack of diversity, which leads to the architecture is non-globally optimal. In this work, the single-level optimization approach is used to optimize the weight of SuperNet and learn the importance of operations in the SuperNet. Instead of sampling from SuperNet randomly, DES assumes that the optimal re-parameterization operation combination vary at different training phases (epochs) and aims to generate different architecture based on current optimal re-parameterization operations. Thus, DES can speed up the convergence of search strategy and explore globally optimal architecture.

2.2 Structural Re-parameterization

The structural re-parameterization technology is an equivalent parameter conversion technology. In our work, the structural re-parameterization technique refers to equivalently converting a multi-branch architecture into a single-branch architecture. ACNet[38] proposes to fuse 1D asymmetric convolution into square convolution to enhance the feature representation capability of square convolution. DDB[26] aims to enhance the representation of a single convolution by combining diverse branches and give methods for fusing multiple convolution operations in various combinatorial forms. RepVGG[27] constructs a residual structure-like branch based on the VGG network and fuses the trained residual-like structure into a 3×3 convolution by structural re-parameterization technique. Based on the re-parameterization technique,

RepNAS[28] designed a re-parameterization search space, in which all multi-branch structures can be transformed into single-branch structures. There are several re-parameterization techniques, which can be described as: 1) *Conv-BN* to *Conv*, 2) a *Conv* for branch addition, 3) *Sequence Conv* structure to *Conv*, 4) a *Conv* for depth concatenation to a *Conv*, 5) *K×K average pooling* to *K×K Conv*, 6) a *Conv* for multi-scale *Convs*. The above work enhances the feature extraction ability of convolution by re-parameterization technology, but their type of re-parameterization operation is deficient. In this work, we aim to build more different types of reparameterization operations.

3 Proposed strategy and search space

Here, the improved re-parameterized search space is designed first. Secondly, the parameters of the SuperNet are optimized by batch optimization method. Afterward we introduce how to generate the offspring architectures. Finally, a re-parameterization verification method is implemented to speed up the verification process.

3.1 Improved re-parameterization search space

The convolution operations in traditional convolution networks are called fixed operation. In this work, fixed operation represents 3×3 convolution operation. In the re-parameterization search space, all candidate operations can be fused into fixed operation. Therefore, when the traditional network (ResNet,VGG etc.) that needs to be re-parameterized is determined, the parameters of the architecture is also determined. In addition, when two different operations are re-parameterized, the center weight of the operations need to be aligned , and then fuse the weight parameters. Hence, the re-parameterization search space has the following characteristics:

1. In structural re-parameterization search space, the parameter number of the network is only related to the number of channels. Therefore, changing the number of operations in the block only affects the resource consumption of training, not the amount of parameters and the inference speed when the network is deployed.
2. In the reparameterization search space, the convolution operations with the same groups and channels but different kernel sizes can be fused into each other, if the centers of the convolutions can be exactly overlapped.

In AcNet[38], taking 3×3 convolution as an example, the cruciform weight of the convolution center position has the most important feature information. Thus, the better feature extraction ability can be achieved by enhancing the cruciform feature at the center position of convolution. In this work, more operations are expected to be included in the re-parameterization search space, which can further enhanced the cruciform weight at the center position of the fixed operation.

Based on the properties of the re-parameterization search space, in this work, $2 \times 2, 2 \times 1$, and 1×2 dilated convolutions are added to the search space, besides the $3 \times 3, 1 \times 3, 3 \times 1, 1 \times 1, 1 \times 1 - 3 \times 3$ convolution operations, residual connection and 1×1-average pooling operation. Since the centers of the 1×2 and 2×1 dilated convolutions overlap with the 3×3 convolution, the dilated convolution can be perfectly fused into 3×3 convolution. Specifically, it can be described as $F^{3 \times 3}_{((0):,:,1:2,::2)} = F^{1 \times 2}_{(D:,:,:,:)}; F^{3 \times 3}_{((0):,:,::2,1:2)} = F^{2 \times 1}_{(D:,:,:,:)}; F^{3 \times 3}_{((0):,:,::2,::2)} = F^{2 \times 2}_{(D:,:,:,:)}$, where $F^{3 \times 3}_{(0)}$ represents 3×3 convolution with weights of zero. $F^{1 \times 2}_{(D)}$, $F^{2 \times 1}_{(D)}$ and $F^{2 \times 2}_{(D)}$ represent the weight of $1 \times 2, 2 \times 1$ and 2×2 dilated convolutions. During re-parameterization, we transfer the weight of dilated convolutions to 3×3 convolution with zero weight. In this way, dilated convolutions can be fused into the fixed operation. Considering that ResNet is one of the most widely used models in visual tasks, in this work, to further improve ResNet through a set of re-parameterization operations, we add a re-parameterization block for 3×3 convolution that is similar to the residual structure.

3.2 Batch optimization of SupNet parameters

In the search process, we use binary encoding (*i.e.*, $0, 1$) to cover the SuperNet to obtain different subnets. 1-element indicates participation in forwarding propagation and 0-element indicates non-participation. To trade off efficiency and accuracy, the method of batch optimization of parameters is introduced, which can be expressed as:

$$d\omega = \frac{1}{P} \sum_{i=1}^{P} d\omega_i = \frac{1}{P} \sum_{i=1}^{P} \frac{\partial L_i}{\partial \omega} \odot \mathcal{M}_i \approx \frac{1}{B} \sum_{j=0}^{B} \frac{\partial L_{\mathcal{M}_j}}{\partial \omega_{\mathcal{M}_j}} \tag{3}$$

where P and B represent the number of populations and subnets. $\mathcal{M}$ denotes the subnet sampled from the SuperNet $\mathcal{S}$. $L_{\mathcal{M}_j}$ is the loss value of the subnet on the training dataset. Eq. 3 shows that the weight parameters of the SuperNet can be optimized by updating the gradients of subnets in batch. Further, it can be approximated as the average gradients of a part of the individuals in the population.

In search process, the branches number of sub-architectures is limited to $\mathcal{C}$ and all sub-architectures share the weight of SuperNet. The single-level method is used to optimize parameters on the training dataset D_{train}. With the formulation used before Eqs. 2 and 3, the search process can be given as:

$$\omega_{t+1}, \theta_{t+1}+ = \xi_{\omega,\theta} \frac{1}{B} \sum_{j=0}^{B} \frac{\partial \mathcal{L}_{D_{train}} (\omega^t, \theta^t)}{\partial \omega_{\mathcal{M}_j}, \theta_{\mathcal{M}_j}} \tag{4}$$

$$\text{s.t.} \begin{cases} \mathcal{M} = \mathcal{S}_{\{\mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_j\}} \\ \|\mathcal{M}_j\| \leq \mathcal{C} \end{cases} \tag{5}$$

where θ and ω are the architecture parameters and the weight parameters of the network, respectively. We generate different sub-architectures to form a population under resource constraints, and optimize the weight and architecture parameters of the Supernet by sampling the architectures from the population.

3.3 Generation of the architecture

The convergence speed of Evolutionary algorithm-based NAS is slow, which is caused by generating offspring architectures randomly. Therefore, we introduce the differentiable method to learn the importance of blocks and re-parameterization operations, and then guide the generation of the sub-architectures. We use the Sigmoid function to quantify the importance of re-parameterization blocks β and candidate operations α. As shown in Fig. 2, each layer of SuperNet is composed of re-parameterization blocks and fixed operations, the block is composed of multiple candidate operations $O_p(\cdot)$. Therefore, the output of the ith layer $\bar{B}^i(x)$ can be expressed as:

$$\bar{B}^i(x) = \beta'_i \sum_{o \in O} \frac{1}{1 + e^{-\alpha_o^i}} f_o(x) + F(x) \tag{6}$$

$$\beta'_i = \frac{1}{1 + e^{-\beta_i}} \tag{7}$$

where $f_o(x)$ and $F(x)$ are the output feature of the candidate operations and fixed operation respectively. From Eqs. 6 and 7, we can conclude the following easily: 1) From a local perspective, the more significant the enhancement of the fixed operation by the re-parameterization operation in a block, the larger the value of α, 2) The feature values of re-parameterization operations are multiplied by the weight β of the current block eventually. From a global perspective, the more important the output feature of the block are for the fixed operation, the greater the value of β. Therefore, when generating offspring architectures, both the global and local characteristics of the architecture are considered.

The architectures in population can be divided into three parts: 1) the architectures retained from the previous population (parent architectures), 2) the offspring architectures generated by the crossover and mutation, 3) the new offspring architectures sampled from the SuperNet. The architectures that are generated according to importance is used to form the third part of the population. Specifically, the random distribution noise σ_α and σ_β are added to the architecture parameters α and β to ensure the diversity of the architecture. We define $\alpha'_0 = \text{sigmoid}(\alpha_0)$, $\alpha' = \text{sigmoid}(\alpha)$, $\beta'_0 = \text{sigmoid}(\beta_0)$, $\beta' = \text{sigmoid}(\beta)$, where α_0 and β_0 are the initialization weights of α and β. The range of the perturbation can be defined as:

$$\sigma_\alpha \in (\alpha'_0 - \max(\alpha'), \alpha'_0 - \min(\alpha')) \tag{8}$$
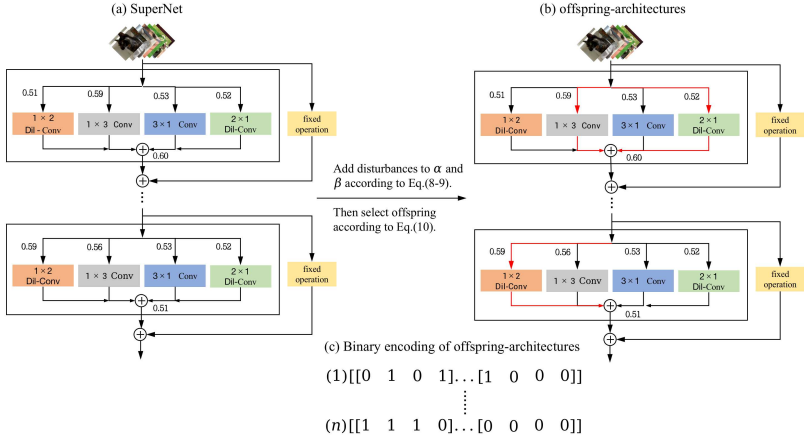
Fig. 2 (a): This is the SuperNet. After training it, each branch and block of the Supernet is given different weights. (b): The offspring architectures are generated from the SuperNet according to Eqs. 8-10. (c): We use binary code to represent the offspring architectures, and the binary code (1) represents the architecture in (b).

$$\sigma_\beta \in \left(\beta_0' - \max\left(\beta'\right), \beta_0' - \min\left(\beta'\right)\right) \tag{9}$$

where σ_α and σ_β belong to random distribution. We take the deviation of the maximum and minimum weight values from the baseline as the range of perturbation. When sampling offspring architectures from the SuperNet, the edges with higher weights are retained by global sorting $(\beta_i' + \sigma_\beta) \cdot (\alpha_i' + \sigma_\alpha)$, $(\cdot)$ denotes the multiplication of two matrices. This process can be simply described as:

$$\begin{cases} 1, & \text{if } \operatorname{rank}\left[(\beta_i' + \sigma_\beta) \cdot (\alpha_i' + \sigma_\alpha)\right] \leq \mathcal{C} \\ 0, & \text{else} \end{cases} \tag{10}$$

where 1-element means the network uses this connection. $rank\,(\cdot)$ denotes the global ranking. As shown in Fig. 2, the red line indicates the branch, which selected, and the black line indicates not selected. Due to the addition of appropriate perturbations, the candidate operations with high weight are retained, and do not completely ignore the operations with low weight in the current stage.
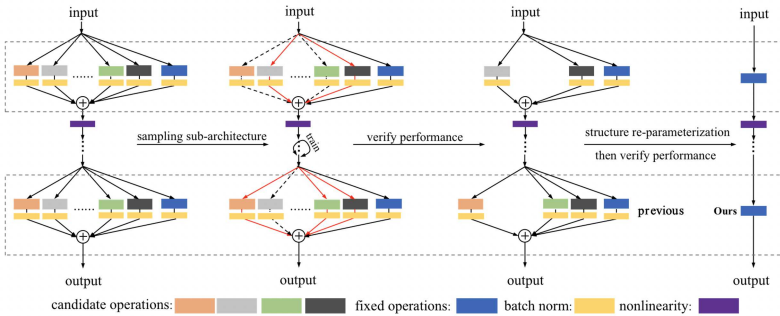

Fig. 3 Using the re-parameterized structure to obtain the accuracy. Each sub-architecture in the population can be re-parameterized into the rightmost structure.

Algorithm 1 Directional evolution strategy for neural architecture search

---

Require: SuperNet $\mathcal{S}$, Population $P = \{P_1, \cdots, P_k\}$, evolution number E_{evo}, Warm up number E_{warm}, parameter optimization epochs E_p, arch-parameters α, β.

1: **while** $i < E_{warm}$ do: **do**
2:      Warm up SuperNet $\mathcal{S}$
3: **end while**
4: **while** $j < E_{evo}$ **do**
5:      **while** $k < E_p$ **do**
6:          **for** Mini-batch data X, target Y in Dataset **do**
7:              Ramdom sample B sub-architectures from Population.
8:              Forward B sampled sub-networks.
9:              Calculate loss and compute the gradients according to Eq. 3.
10:              Update network parameters ω and architecture parameters α, β.
11:          **end for**
12:      **end while**
13:      Re-parameterize architecture and obtain the performance of the architecture.
14:      Select the architecture according to performance and perform mutation crossover. Meanwhile, sample the subarchitectures from the SperNet according to Eq. 8-10.
15: **end while**
16: Output: Architectures $P = \left\{ P_1^{'}, P_2^{'} \cdots P_k^{'} \right\}$.

3.4 Performance estimation of population

In evolutionary algorithm-based NAS, evaluating the architectures takes a lot of time. In addition, there is still tiny deviation in the performance of the architecture before and after reparameterization. Although this deviation can be ignored in practical application, we need to search for the best performance architecture after re-parameterization, it has a certain impact on the choice of the architecture. Thus, in this work, the re-parameterization operations are fused into the fixed operation before verifying architecture performance, as shown in Fig. 3. It is worth noting that the *BN* layer is also fused into the fixed convolution. The multi-branch *Conv-BN* layer becomes *Conv* layer. Hence, using re-parameterized architectures for performance evaluation can speed up the evaluation process and eliminate the deviation.

We use α and β to indicate the importance of blocks and candidate operations. Therefore, the weights and biases of the candidate operations need to be scaled $\alpha \cdot \beta$ times before the architecture is re-parameterized. Table 1 shows the time consumption of the verification process on CIFAR-10. Concretely, the number of populations is set to [64, 128, 256]. Our approach increases the speed of the architecture evaluation by around 60% compared to the naive evaluation method. In this experiment, considering factors such as architecture diversity search time and computational resources, the population size is set to 128. The overall training procedure is summarized in Algorithm 1.

4 Experiments

To verify the improved re-parameterization search space is effective for different datasets, we search for a set of reparameterization operations for ResNet on CIFAR-10 and ImageNet-1K. Due to ResNet has residual structure, it is natural to weed out residual connection in the search space. Our experiment is divided into two stages: the search stage and the retraining stage.

4.1 Search Architectures on CIFAR-10

We add a re-parameterization block similar to the residual structure to the 3×3 convolution in VGG-16 and ResNet-18. For a fair comparison, the structure of the network and data augmentation techniques are followed by ACNet[38] and ResNet[18]. We use the SGD optimizer with a learning rate of 0.1 to optimize the parameters of the network. To optimize the architecture parameters α and β, we use Adam optimizer with learning rate of 0.0001 and $(0.5, 0.999)$ betas. We limit the number of branches to $\frac{2}{3}$ times of the total branch number. In the training process, we sampled 5 architectures and used them to update SuperNet. The probability of both mutation and crossover for the architecture is 0.5.

We search for 500 epochs and retrain architectures on CIFAR-10 dataset. Except for the learning rate and the probability of the drop-path, the retraining process are the same as DARTS[1]. Respectively, the learning rate and the probability of drop-path is set to 0.05 and 0.08.

Table 2 shows our results. We achieved 1.02% better accuracy than RepVGG[27] and 0.21% than RepNAS[28]. Our architecture has a great advantage in the inference process. Since the re-parameterized architecture retains only convolution and non-linear operations, the inference speed of IrepResNet-18 and IrepVGG-16 reaches 3.93ms and 1.76ms per image, which is faster than the architectures such as DARTS.

Table 1 We obtain the performance of architecture on Nvidia A100 GPU. The time to evaluate the performance of the architecture by the re-parameterization techniques consists of two parts, i.e. the time consumed by the architecture re-parameterization and the architecture forward inference. The results are average of verifying 10 populations and the batch size is 512, full precision(fp32).

Population Size/Time		64/S	128/S	256/S
Multi-Branch	VGG-16	1349.4	2681.1	5336.9
	Resnet-18	649.1	1301.6	2685.7
Re-parameterized	VGG16	499.5	1000.0	1823.9
	Resnet-18	286.3	571.5	1133.9
Acceleration percentage (%)	VGG-16	63.7	62.7	65.8
	Resnet-18	55.9	56.1	57.8

4.2 Experience on ImageNet-1K

To reveal the generalization ability, we evaluate on the ImageNet-1K, which contains 1.3M images for training and 50K for validation from 1000 classes. To save computational resource and speed up the search, based on the conclusion of ACNet[38], we remove the 2×2 dilated convolution from the search space. We set $B = 1$, $E_{warm} = 5$ and batch size is 256. We use Adam optimizer with 0.0001 learning rate and (0.5, 0.999) betas to optimize α, β. We limit the number of branches to $\frac{1}{2}$ times of the total branch number. We search for 100 epochs and then fix the structure of the SuperNet to retrain architectures for 120 epochs. To be fair, we use the same data augmentation techniques as ResNet[18].

We compare our architectures with state-of-the-arts in Table 3. Compared to other work, IrepResNet also shows favorable performance. IrepResNet-50 achieve top-1 accuracy of 77.92%, which is 1.82% higher than ResNet-50, 0.84% higher than DyRep[29] and 1.21% higher than DDB[26]. IrepResNet-34 and IrepResNet-18 also achieve great performance. Meanwhile, the architecture has faster inference speed compared to ResNet, DARTS, P-DARTS, GoldNAS, etc.

We plot the structure of IrepResNet in Appendix A. The structure of IrepResNet-50 is truncated in the middle, i.e., the first eight layers retain all re-parameterization operations, and the last eight layers exclude all enhancement operations. To better explain this phenomenon, we visualized the output feature values of ResNet-50 and IrepResNet-50. As shown in Fig. 4, it can be easily concluded that IrepResNet-50 has stronger discrimination ability for targets compared to ResNet-50. For ResNet-50, the role of the first eight layers task is mainly to achieve the separation of foreground and background in the image. While the last eight layers task is mainly to further distinguish the subtle differences between foreground and background, so that the target in the image can be focused accurately. This division of task is significant for the formation of the IrepResNet-50 structure. The first eight layers obtain less feature

Table 2 Comparison with state-of-the-art image classifiers on CIFAR-10 dataset. We calculated the parameters of the model and tested the model of the inference time on Nvidia A100 GPU with a batch size of 1, full precision(fp32).

Model	Top-1 (%)	Params(M)	Inference (ms)	Search Cost (GPU days)
VGG	94.12	14.73	2.14	—
ResNet-18	96.21	11.69	4.12	—
RepVGG	94.62	14.73	1.76	—
RepNAS (VGG)	95.43	11.69	1.76	0.7
AcNet	94.47	14.73	1.76	—
DARTS (second)	97.24	3.3	31	4
P-DARTS	97.50	3.4	33	0.3
GoldNAS	97.39	3.67	40	1.1
CARS	97.38	3.6	27	0.4
AmoebaNet-A	96.60	3.2	38	3150
IrepResNet-18	**96.61**	**11.69**	**3.93**	**8.0**
IrepVgg-16	**95.65**	**14.73**	**1.76**	**16.0**

Table 3 Results of our models on ImageNet-1K dataset compared to other models. The performance on ImageNet-1K with comparison to other NAS methods and models. All experiments on the ImageNet-1K were performed based on Nvidia A100 GPU. We calculated the parameters of the model and tested the model of the inference time with a batch size of 1, full precision(fp32).

Model	Top-1(%)	Top-5(%)	Params(M)	Inference (ms)	Search Cost (GPU-days)
ResNet18	69.76	89.07	11.69	4.25	—
ResNet34	73.31	91.42	21.80	6.12	—
ResNet50	76.10	93.29	25.56	7.54	—
DyRep (ResNet-18)	71.58	—	16.90	3.59	—
DyRep (ResNet-34)	74.68	—	33.10	5.15	—
DyRep (ResNet50)	77.08	—	31.50	6.14	—
DDB (ResNet18)	70.99	—	26.30	3.59	—
DDB (ResNet34)	74.33	—	49.90	5.15	—
DDB (ResNet50)	76.71	—	40.70	6.14	—
DARTS (second)	73.30	91.3	4.70	67.4	4
P-DARTS	75.60	92.6	4.90	62.3	0.3
GoldNAS	76.10	92.7	6.40	39.4	1.7
CARS	75.20	92.5	5.10	59.9	0.4
IrepResNet-18	**71.57**	**89.98**	**18.23**	**3.59**	**12**
IrepResNet-34	**74.91**	**92.12**	**36.12**	**5.15**	**19**
IrepResNet-50	**77.92**	**93.88**	**29.04**	**6.14**	**30**

information due to the few number of channels. Thus, the re-parameterization operations are important to improve the current feature information, which makes the them has higher weight. The last eight layers can acquire more feature information because of the more channels, which allows it to take on the task of focusing on the target and refining the foreground and background in the picture. Compared with the first eight re-parameterization blocks, the feature information in the last eight re-parameterization blocks is not important for the last eight fixed operations, which makes the re-parameterization operations has smaller weight value. Therefore, searching the architecture under resource constraints makes the algorithm prefer to retain the re-parameterization operations that in the shallow layers. We also visualized the output features and architecture of IrepResNet-34 and IrepResNet-18, as shown in Appendix A. The re-parameterization operations that are preserved in the IrepResNet-34 and IrepResNet-18 also emerge the same trend.

4.3 Generalization performance on Downstream Task

We transfer our ImageNet-pretrained IrepResNet-50 and IrepResNet-18 model to downstream tasks object detection to validate generalization of the model. Specifically, the pre-trained model is used as the backbone for the downstream algorithms FPN[39] and CenterNet[40] algorithms on the COCO dataset. For the optimization of the target detection model, we refer to the optimization
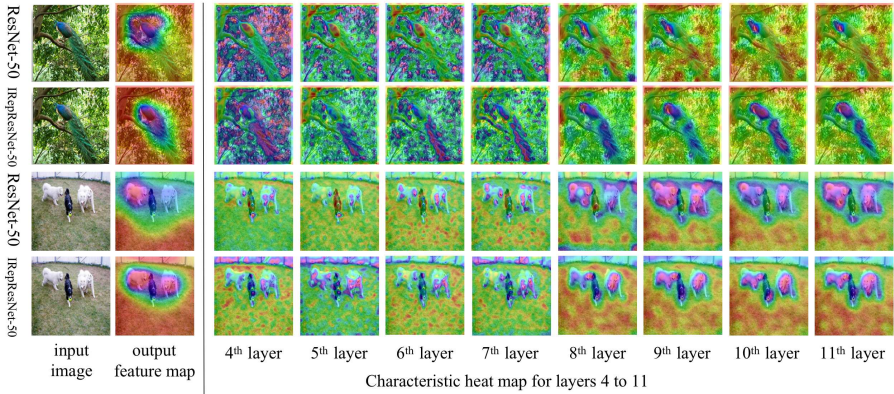
Fig. 4 We visualized the output feature values of the convolution in the ResNet-50 and IRepResNet-50 to better interpret the structure of our network. We consider the structure of Conv1×1-Conv3×3-Conv1×1 as a layer. The first and second columns are the original image and the heatmap of the last layer output. The other columns are the heatmap of the feature outputs convolved in the 4th layer to the 11th layer. IRepResNet-50 has significantly better feature focus than ResNet-50.

Table 4 Results on object detection.

Backbone	Algorithm	ImageNet Top-1	COCO mAP
ResNet-18	CenterNet	69.76	29.5
ResNet-50	FPN	76.10	37.9
DyRep-ResNet50	FPN	77.08	38.1
Irep-ResNet18	CenterNet	71.27	**31.2**
Irep-ResNet50	FPN	77.92	**38.2**

Table 5 Performance of the architecture on the CIFAR-10 dataset under different resource constraints.

Model	1/6	1/3	1/2	2/3	5/6	1
IRepVGG-16	94.22	94.58	95.21	95.65	95.64	95.64
IRepResNet-18	96.28	96.47	96.47	96.61	95.50	96.52

approach and hyperparameter settings of MMDetection[41]. FPN and Center-Net are fine-tuned on a single NVIDIA A100 GPU with batch sizes 16 and 64, respectively. In addition, the fine-tuned model can re-parameterize the backbone to achieve faster forward inference. The results in Table 4 show that IrepResNet can achieve better performance compared to FPN, CenterNet, and DyRep.
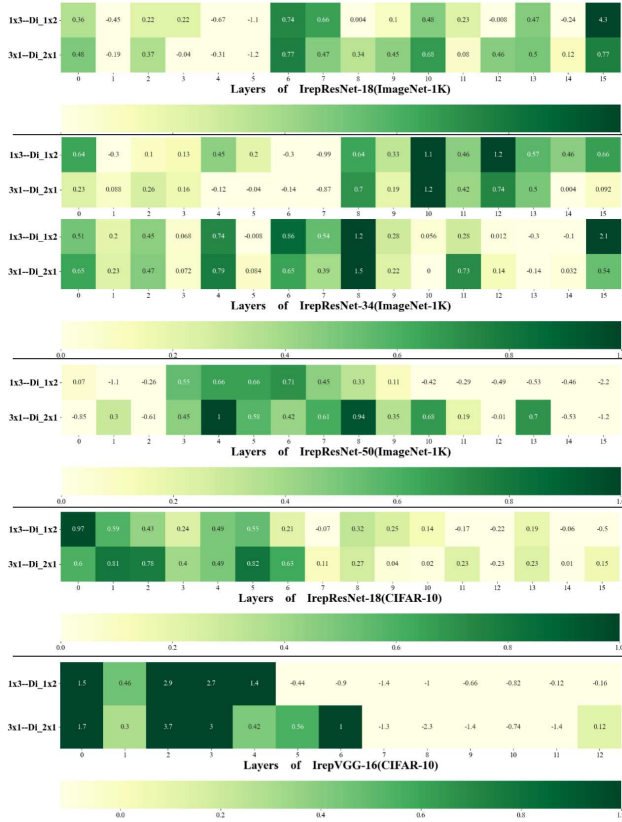
Fig. 5 Differential heatmap of the 1×3 convolution–1×2 dilated convolution and 3×1 convolution–2×1 dilated convolution, which searched on CIFAR-10 and ImageNet-1K dataset

4.4 Ablatilaon Study

4.4.1 Search under different resource constraints

To explore the impact on the performance of the architecture under different resource constraints, we searched IrepVGG-16 and IrepResNet-18 on the CIFAR-10 dataset. Specifically, the number of branches is set to $\left[\frac{1}{6}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{5}{6}, 1\right]$ times of the total branch number. As shown in Table 5, when the resource constraint reaches $2/3$, the architecture achieves better performance. We found that the performance of the architecture is weaker than RepNAS[28] when retaining the same number of branches as RepNAS in the improved reparameterization search space (retain four branches for each block). Based on AcNet[38], the main reason is that the enhancement effect of the dilated convolutions on the 3×3 convolution is weaker than the 1×3 and 3×1 convolutions. As shown in Fig. 5, the architecture weights of the 1×3 convolution–1×2 dilated convolution and 3×1 convolution–2×1 dilated convolution are subtracted and transformed equivalently and we plotted them as

heatmap. The larger the difference values, the more important the asymmetric convolution (3×1 and 1×3 convolutions) in the same layer. This indicates that the feature enhancement effect of asymmetric convolutions (1×3 and 3×1 convolution) are stronger than dilated convolutions in this experiment. Therefore, when the same number of branches are retained as RepNAS[28], some of the 1×3 and 3×1 convolutions may be replaced by dilated convolution due to the Matthew effect of the gradient-based learning method, which leads to the architectures with potentially weaker performance than RepNAS[28].

5 Conclusion

In order to further improve tranditional convolutional networks, we designed a more comprehensive re-parameterization search space and searched it by directional evolutionary strategy to further improve the performance of ResNet. A re-parameterization block similar to the residual connection is added to each 3×3 convolution in ResNet model. Then finding an optimal architecture population by exploring the derivative architectures of the optimal re-parameterization architecture at the current stage. Extensive experiments demonstrate that the proposed the improved re-parameterization search space can further improve the performance of models and perform well in downstream tasks. Moreover, we explain the reasons for the formation of the architecture and analyze the enhancement effect between re-parameterization operations. It is worth mentioning that the improved re-parameterization search space proposed in our paper can be further used as a bridge between coarse-grained search and fine-grained search. It means that the re-parameterization model after coarse-grained search (architecture operation) can be divided into 1×1 convolution and $2 \times 2, 2 \times 1, 1 \times 2$ dilated convolution, and then the model can be transformed into a channel pruning friendly network, which can actually further reduce the FLOPs and inference time of the model.
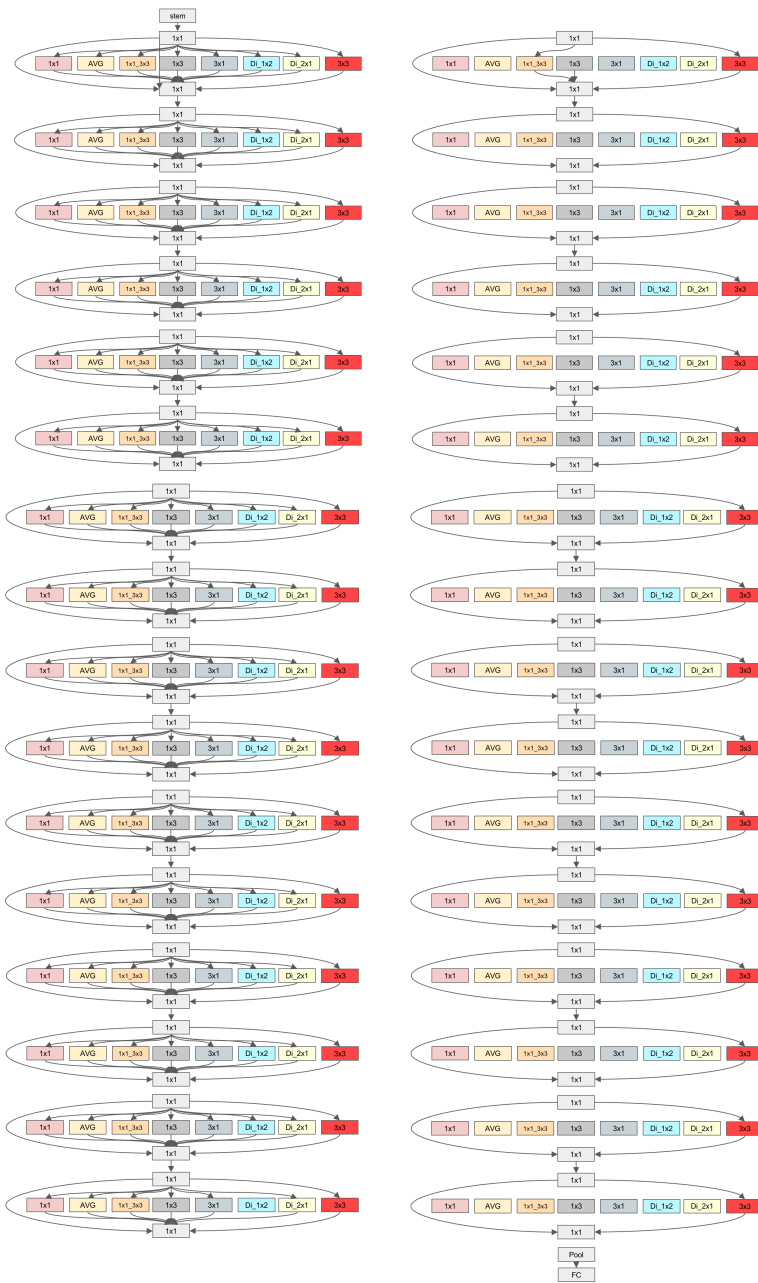
Appendix A   IrepResNet model

Fig. A1 IrepResnet-50 searched on ImgeNet. The 3×3 convolution operation is a fixed operation and does not participate in the search process of the architecture.

Fig. A2 List of architectures of IrepResnet-50, IrepResnet-18, IrepResnet-34. Each row represents the enhancement of a 3×3 convolution. From left to right, it represents the re-parameterization structure from the first layer to the last layer.
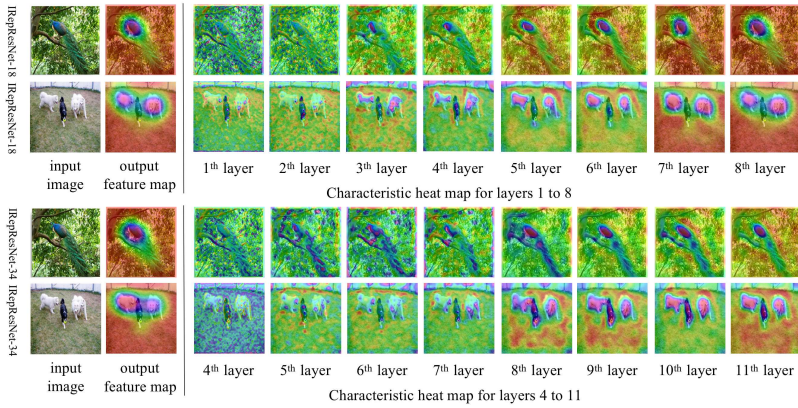

Fig. A3 We visualized the output feature values of the convolution in the IrepResNet-18 and IrepResNet-34 to better interpret the structure of our architecture. We consider the structure of Conv3 × 3-Conv3 × 3 as one layer.

References

[1] Liu, H., Simonyan, K., Yang, Y.: Darts: Differentiable architecture search. arXiv preprint arXiv:1806.09055 (2018)

[2] Hu, S., Xie, S., Zheng, H., Liu, C., Shi, J., Liu, X., Lin, D.: Dsnas: Direct neural architecture search without parameter retraining. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12084–12092 (2020)

[3] Cai, H., Zhu, L., Han, S.: Proxylessnas: Direct neural architecture search on target task and hardware. arXiv preprint arXiv:1812.00332 (2018)

[4] Xie, S., Zheng, H., Liu, C., Lin, L.: Snas: stochastic neural architecture search. arXiv preprint arXiv:1812.09926 (2018)

[5] Lu, Z., Whalen, I., Boddeti, V., Dhebar, Y., Deb, K., Goodman, E., Banzhaf, W.: Nsga-net: neural architecture search using multi-objective genetic algorithm. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 419–427 (2019)

[6] Chu, X., Zhou, T., Zhang, B., Li, J.: Fair darts: Eliminating unfair advantages in differentiable architecture search. In: European Conference on Computer Vision, pp. 465–480 (2020). Springer

[7] Yang, Z., Wang, Y., Chen, X., Shi, B., Xu, C., Xu, C., Tian, Q., Xu, C.: Cars: Continuous evolution for efficient neural architecture search. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1829–1838 (2020)

[8] Guo, Z., Zhang, X., Mu, H., Heng, W., Liu, Z., Wei, Y., Sun, J.: Single path one-shot neural architecture search with uniform sampling. In: European Conference on Computer Vision, pp. 544–560 (2020). Springer

[9] Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 779–788 (2016)

[10] Redmon, J., Farhadi, A.: Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767 (2018)

[11] Liu, C., Chen, L.-C., Schroff, F., Adam, H., Hua, W., Yuille, A.L., Fei-Fei, L.: Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 82–92 (2019)

[12] Fang, J., Sun, Y., Zhang, Q., Peng, K., Li, Y., Liu, W., Wang, X.: Fna++: Fast network adaptation via parameter remapping and architecture search. IEEE Transactions on Pattern Analysis and Machine Intelligence **43**(9), 2990–3004 (2020)

[13] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)

[14] Zhu, J.-Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image

translation using cycle-consistent adversarial networks. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2223–2232 (2017)

[15] Li, W., Zhu, X., Gong, S.: Person re-identification by deep joint learning of multi-loss classification. arXiv preprint arXiv:1705.04724 (2017)

[16] Xiao, T., Li, S., Wang, B., Lin, L., Wang, X.: Joint detection and identification feature learning for person search. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3415–3424 (2017)

[17] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)

[18] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)

[19] Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)

[20] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.-C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4510–4520 (2018)

[21] Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., *et al.*: Searching for mobilenetv3. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 1314–1324 (2019)

[22] Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4700–4708 (2017)

[23] Wan, A., Dai, X., Zhang, P., He, Z., Tian, Y., Xie, S., Wu, B., Yu, M., Xu, T., Chen, K., *et al.*: Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12965–12974 (2020)

[24] Hong, W., Li, G., Zhang, W., Tang, R., Wang, Y., Li, Z., Yu, Y.: Dropnas: Grouped operation dropout for differentiable architecture search. arXiv preprint arXiv:2201.11679 (2022)

[25] Xu, Y., Xie, L., Zhang, X., Chen, X., Qi, G.-J., Tian, Q., Xiong, H.: Pc-darts: Partial channel connections for memory-efficient architecture search. arXiv preprint arXiv:1907.05737 (2019)

[26] Ding, X., Zhang, X., Han, J., Ding, G.: Diverse branch block: Building a convolution as an inception-like unit. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10886–10895 (2021)

[27] Ding, X., Zhang, X., Ma, N., Han, J., Ding, G., Sun, J.: Repvgg: Making vgg-style convnets great again. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 13733–13742 (2021)

[28] Zhang, M., Yu, X., Rong, J., Ou, L., Gao, F.: Repnas: Searching for efficient re-parameterizing blocks. arXiv preprint arXiv:2109.03508 (2021)

[29] Huang, T., You, S., Zhang, B., Du, Y., Wang, F., Qian, C., Xu, C.: Dyrep: Bootstrapping training with dynamic re-parameterization. arXiv preprint arXiv:2203.12868 (2022)

[30] Lu, Z., Deb, K., Goodman, E., Banzhaf, W., Boddeti, V.N.: Nsganetv2: Evolutionary multi-objective surrogate-assisted neural architecture search. In: European Conference on Computer Vision, pp. 35–51 (2020). Springer

[31] Chen, Y., Meng, G., Zhang, Q., Xiang, S., Huang, C., Mu, L., Wang, X.: Renas: Reinforced evolutionary neural architecture search. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4787–4796 (2019)

[32] Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y., Keutzer, K.: Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10734–10742 (2019)

[33] Yang, Y., You, S., Li, H., Wang, F., Qian, C., Lin, Z.: Towards improving the consistency, efficiency, and flexibility of differentiable neural architecture search. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6667–6676 (2021)

[34] Bi, K., Xie, L., Chen, X., Wei, L., Tian, Q.: Gold-nas: Gradual, one-level, differentiable. arXiv preprint arXiv:2007.03331 (2020)

[35] Li, G., Qian, G., Delgadillo, I.C., Muller, M., Thabet, A., Ghanem,

B.: Sgas: Sequential greedy architecture search. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1620–1630 (2020)

[36] Yang, Y., You, S., Li, H., Wang, F., Qian, C., Lin, Z.: Towards improving the consistency, efficiency, and flexibility of differentiable neural architecture search. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6667–6676 (2021)

[37] Hou, P., Jin, Y.: Single-level optimization for differential architecture search. arXiv preprint arXiv:2012.11337 (2020)

[38] Ding, X., Guo, Y., Ding, G., Han, J.: Acnet: Strengthening the kernel skeletons for powerful cnn via asymmetric convolution blocks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 1911–1920 (2019)

[39] Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2117–2125 (2017)

[40] Zhou, X., Wang, D., Krähenbühl, P.: Objects as points. arXiv preprint arXiv:1904.07850 (2019)

[41] Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., et al.: Mmdetection: Open mmlab detection toolbox and benchmark. arxiv 2019. arXiv preprint arXiv:1906.07155 (2019)