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 · Linlin Ou 1

Accepted: 5 February 2023 / Published online: 10 March 2023
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

Abstract
Traditional NAS methods improve performance by sacrificing the landing ability of the architecture and the re-parameterization technology is expected to solve this problem. However, most current Rep methods rely on prior knowledge to select the re-parameterization operations, which limits the architecture performance to the type of operations and prior knowledge. At same time, some re-parameterization operations hinder the optimization of the network. To break these restrictions, in this work, an improved re-parameterization search space is designed, including more type of re-parameterization operations. Concretely, the performance of convolutional networks can be further enhanced by the search space. An automatic re-parameterization enhancement strategy is designed to effectively explore this search space based on neural architecture search (NAS), which can search an excellent re-parameterization architecture. Then, we solved the optimization problem caused by using some re-parameterization operations to enhance ResNet-style network. 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. Still, some convolution operations (such as $7 \times 7$ and $5 \times 5$ separable convolutions, $7 \times 1$–$1 \times 7$ sequential convolution) in the search space are not well optimized on the terminal devices, which makes them [2–4, 6, 23–26] not well deployed on edge devices. In Fig. 1, we tested network accuracy and inference time 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 traditional networks. Still, the specific operations need to be selected based on prior knowledge [27, 28], 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 model performance and find the globally optimal networks, we use the NAS method to search for a set of re-parameterization operations that can be fused into the traditional networks. In this way, we can get excellent 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 [29, 30] searches the best combination of re-parameterization operations automatically, but the type of re-parameterization operations limits the upper limit of network performance. Therefore, the model performance can be further improved by exploring more re-parameterization operations.

![Fig. 1](image)

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.
In addition to the re-parameterization operations involved in the above work [29, 30], there are other convolution operations that can be fused into VGG-style or ResNet-style networks after appropriate transformation. Thus, an improved re-parameterization search space (IREPS) was designed to include more re-parameterization operations, which can be used for architecture search and unstructured pruning. 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. However, some re-parameterization operations (such as residual operation) bring optimization problem to ResNet-style networks. Therefore, in order to solve the problem, we also proposed a method to build re-parameterization blocks for ResNet-style networks.

For the larger re-parameterization search space, a directional evolutionary strategy (DES) was designed to explore an 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, making the algorithm rapidly converge. To explain the reasons for the formation of the architecture and the improvement of performance, we visualized the network structure and its output feature. In summary, the contributions can be summarized as follows:

1. An improved re-parameterization search space is designed, which contains more re-parameterization operations. Compared with other Rep search space, it can further enhance the performance of the traditional convolutional networks. In addition, it can also be used to achieve unstructured pruning of the network.
2. We prove that the residual operation in the re-parameterization search space increases the optimization difficulty of the ResNet-style networks when searching for the re-parameterization architecture and propose a method to solve it.
3. To explore the larger search space, DES is proposed to search a set of re-parameterization operations. It can balance the global and local feature of the re-parameterization architecture, and explore a group of re-parameterization operations with the best feature enhancement effect.
4. We explain the reason for the formation of the re-parameterization architecture. And extensive experiments on image classification and its downstream tasks demonstrate that our architecture achieves better results than other related work.

2 Related Work

2.1 Network Architecture Search

Neural architecture search (NAS) is a widely-used technique that aims to search feature extraction networks that match given tasks. Evolutionary algorithm-based NAS [5, 7, 8, 31, 32] uses the principle of "survival of the fittest" to select architectures and relies on genetics, mutation, crossover, and random generation to obtain new offsprings. The evolutionary algorithm is a 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, 26, 33–37] 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 \[1, 6, 34, 36, 37\], which optimizes the architecture parameters under optimal weight parameters. It can be described as:

\[
\min_{\alpha} L_{val}(w^*(\alpha), \alpha) \quad \text{s.t.} \quad w^*(\alpha) = \arg\min_w L_{\text{train}}(w, \alpha)
\]

(1)

where \(\alpha\) denotes architecture, \(w_\alpha\) denotes the network weight bound with the architecture \(\alpha\), \(L_{\text{train}}\) and \(L_{val}\) denote optimization loss on training and validation dataset. It first optimizes the network weights \(w\), then finds \(\alpha\) that minimizes the validation loss \(L_{val}\). The other is single-level optimization \[2, 38\], which regards the optimization of \(w\) and \(\alpha\) as independent processes. It can be described as:

\[
\alpha^t, w^t = \eta \nabla_{\alpha, w} L_{\text{train}}(\alpha^{t-1}, w^{t-1})
\]

(2)

Eq. (2) indicates that both \(w\) and \(\alpha\) are optimized in an optimization process. Although gradient-based NAS can converge quickly, Matthew effect makes architecture lack diversity, which makes the architecture 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 SuperNet. Instead of sampling from SuperNet randomly, DES assumes that the optimal re-parameterization operation combination varies at different training phases 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.

\subsection{2.2 Structural Re-parameterization}

Structural re-parameterization technology is an equivalent parameter conversion technology. In our work, the structural re-parameterization technique equivalently converts a multi-branch architecture into a single-branch one. ACNet \[39\] proposes to fuse 1D asymmetric convolution into square convolution to enhance the feature representation capability of square convolution. DDB \[27\] aims to enhance the representation of a single convolution by combining diverse branches and given methods for fusing multiple convolution operations in various combinatorial forms. RepVGG \[28\] constructs a residual structure-like branch based on the VGG network and fuses the trained residual-like structure into a \(3 \times 3\) convolution by structural re-parameterization technique. Based on the re-parameterization technique, RepNAS \[29\] designed a re-parameterized 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) \(K \times K\) averagepooling to \(K \times K\) 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 re-parameterization operations.

\section{3 Proposed Strategy and Search Space}

Here, the improved re-parameterized search space is designed first. Secondly, SuperNet is encoded by the 0–1 encoding method and the parameters of the SuperNet are optimized by the batch optimization method. Afterwards, 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

In this section, we further research the re-parameterization search space and summarize its characteristics. On this basis, we design new re-parameterization operations to expand the re-parameterization search space.

In this work, the convolution operations in traditional convolution networks are called fixed operations. Specifically, fixed operation represents $3 \times 3$ convolution in ResNet model. All candidate operations can be fused into fixed operations in the re-parameterization search space. Therefore, when the traditional network (ResNet, VGG, etc.) needs to be re-parameterized is determined, the architecture’s parameters are also determined. In addition, when two different operations are re-parameterized, the center weight of the operations needs 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 number of parameters and the inference speed when the network is deployed.

2. In the reparameterization search space, convolution operations with the same groups and channels but different kernel sizes can be fused into each other if the convolutions centers can be exactly overlapped.

In AcNet [39], taking $3 \times 3$ convolution as an example, the cross-shaped weight at the center position of convolution has the most crucial feature information. Thus, a better feature extraction ability can be achieved by enhancing the cross-shaped 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 enhance the fixed operation.

Based on the properties of the re-parameterization search space, in this work, $2 \times 2$, $2 \times 1$, and $1 \times 2$ 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 $1 \times 1$-average pooling operation, as shown in Table 1. It summarizes the detailed operation spaces. Compared with related works, we have more candidate operations. In the re-parameterization search space, each block retains at least one candidate operation. The size of the re-parameterization search space can be expressed as: $N\sum_{i=1}^{B} C_i$, where $B$ represents the number of candidate operations and $N$ is the number of fixed operations that stacked in the blocks. Therefore, there are $1023^N$ different re-parameterization architectures in our search space, and it is $8^N$ times that of RepNAS [29] and DyRep [30].

Since the centers of the $1 \times 2$ and $2 \times 1$ dilated convolutions overlap with the $3 \times 3$ convolution, the dilated convolution can be perfectly fused into the $3 \times 3$ convolution. Specifically, it can be described as $F^{3 \times 3}_{((0,:)\cdots;12,:)2} = F^{1 \times 2}_{(D,:)\cdots;}$; $F^{3 \times 3}_{((0,:)\cdots;2,1,:)2} = F^{2 \times 1}_{(D,:)\cdots;}$; $F^{3 \times 3}_{((0,:)\cdots;2,:)2} = F^{2 \times 2}_{(D,:)\cdots;}$, where $F^{3 \times 3}_{(D,:)\cdots;}$ represents $3 \times 3$ 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 $2 \times 2$ dilated convolutions. During re-parameterization, we cover the weight of dilated convolutions to $3 \times 3$ 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,
Table 1  Operation spaces of related works

| Method    | Branches | Candidate operation | Search space size |
|-----------|----------|---------------------|-------------------|
| RepVGG    | 3        | $K \times K, 1 \times 1, \text{residual connection}$ | —                 |
| DBB       | 4        | $K \times K, 1 \times 1-K \times K, 1 \times 1-\text{AVG}, 1 \times 1$ | —                 |
| RepNAS    | 7        | $K \times K, 1 \times 1-K \times K, 1 \times 1-\text{AVG}, 1 \times 1, 1 \times K, K \times 1, \text{residual connection}$ | $127^N$           |
| DyRep     | 7        | $K \times K, 1 \times 1-K \times K, 1 \times 1-\text{AVG}, 1 \times 1, 1 \times K, K \times 1, \text{residual connection}$ | $127^N$           |
| Irep(Ours)| 10       | $K \times K, 1 \times 1-K \times K, 1 \times 1-\text{AVG}, 1 \times 1, 1 \times K, K \times 1, \text{Di-} \frac{K+1}{2} \times \frac{K+1}{2}, 1 \times \frac{K+1}{2}, \frac{K+1}{2} \times 1, \frac{K+1}{2} \times \frac{K+1}{2}, \text{residual connection}$ | $1023^N$          |

$K \times K$ denotes Conv operation with the kernel size $K \times K$, and $1 \times 1-K \times K$ denotes a branch stacking $1 \times 1$ and $K \times K$ Conv sequentially. Di-$\frac{K+1}{2} \times \frac{K+1}{2}$ denotes dilated Conv operation with the kernel size $\frac{K+1}{2} \times \frac{K+1}{2}$ in re-parameterization search space.

we add a re-parameterization block for $3 \times 3$ convolution similar to the residual structure further to improve ResNet through a set of re-parameterization operations.

In addition, as shown in Fig. 2, we can split a $3 \times 3$ convolution (square convolution) into $1 \times 1$ convolution, $1 \times 2$ dilated convolution, $2 \times 1$ dilated convolution, and $2 \times 2$ dilated convolution by the proposed search space. Then for a network with VGG-style or ResNet-style that has been pre-trained, we can import the pre-training weight of $3 \times 3$ convolution into four independent branching operations to achieve pruning in the spatial dimension.

Although the re-parameterization search space can enhance the performance of traditional convolutional networks, we find that the residual operation in the search space make it difficult to optimize the ResNet-style networks (mathematical proof see Appendix A.2). Therefore, for pure VGG-style networks, we suggest adding residual operation in a ResNet-like manner (2 layers per block). As for the ResNet-style network with residual structure, we propose to remove the residual operation from the re-parameterization search space. Although the size of the re-parameterization search space is reduced, it ensures the optimization effect of the networks.

3.2 Encoding Method and Batch Optimization of SupNet Parameters

This section presents SuperNet encoding to obtain subnets and then proposes batch optimization method to optimize SuperNet parameters.

In the search process, we adopt 0–1 encoding method to cover the SuperNet to obtain different subnets. 1-element indicates participation in forward propagation, and 0-element..
Fig. 3 The way of encoding IrepResNet and the process of crossover and mutation. $K \times K$ represents square convolution, which is fixed in the SuperNet, and does not participate in the encoding process. The c, d, e, and f represent re-parameterization operations. Parent architectures generate offspring by the selection, mutation, and crossover operations.

indicates non-participation. As shown in Fig. 3, the output feature value of the candidate operation is multiplied by corresponding 0–1 element to obtain the final output. When the feature value is multiplied by 0, the operation can be regarded as non-existent, whether in forward or backward propagation. Multiple groups of 0–1 element are adopted to cover all the blocks of SuperNet and regard the matrix composed of these 0–1 elements as the subnets selected from SuperNet. In this experiment, we fix the position of candidate operations. Therefore, the 0–1 encoding method is more concise and intuitive than the real encoding method to reflect the structure of re-parameterization. Moreover the 0–1 element directly participates in the propagation process of architecture, which reduces the occupation of computational memory and the consumption of computational resources.

We take the verification accuracy of the architectures as their fitness and use simple mutation method to randomly mutate 0–1 element through a certain probability. When crossing genotypes, we adopt the one-point crossover method to cross the genotypes and regard the structure in a block as the most basic unit. By randomly selecting the crossover position and setting the crossover probability to achieve the genotypes crossover of two-parent architectures.
To trade off efficiency and accuracy, the method of batch optimization of parameters is introduced, which can be expressed as:

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

(3)

where \( P \) and \( B \) represent the number of populations and subnets. \( \mathcal{M} \) denotes the subnet selected from the SuperNet \( \mathcal{S} \). \( L_{\mathcal{M}_j} \) is the loss value of the subnet on the training dataset. Equation (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 individuals in the population.

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

\[
\omega_{t+1}, \theta_{t+1} = \xi_{\omega, \theta} \frac{1}{B} \sum_{j=0}^{B} \frac{\partial L_{D_{\text{train}}}}{\partial \omega_{\mathcal{M}_j}, \theta_{\mathcal{M}_j}}
\]

(4)

\[\text{s.t.} \quad \mathcal{M} = \mathcal{S}\{\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_j\} \quad \|\mathcal{M}_j\| \leq C \]

(5)

where \( \theta \) and \( \omega \) are the architecture and the weight parameters of network, 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

In this section, a method for generating subnets is proposed. Then the advantages of this method are presented from the global and local perspectives.

Evolutionary algorithm-based NAS convergence speed is slow, 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 adopt the Sigmoid function to quantify the importance of re-parameterization blocks \( \beta \) and candidate operations \( \alpha \). As shown in Fig. 4, 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 \( i^{th} \) layer \( \bar{B}^i(x) \) can be expressed as:

\[
\bar{B}^i(x) = \beta^i_{o} \sum_{o \in O} \frac{1}{1 + e^{-\alpha^i_o}} f_o(x) + F(x)
\]

(6)

\[
\beta^i_{o} = \frac{1}{1 + e^{-\beta_i}}
\]

(7)

where \( f_o(x) \) and \( F(x) \) are the output feature of the candidate operations and fixed operation respectively. From Eqs. (6)–(7), we can conclude the following easily: (1) From a local perspective, the parameter \( \alpha \) enables the re-parameterization operation to focus on the feature extraction ability of network blocks and enhances the fixed operation as much as possible by selecting appropriate re-parameterization operations. (2) From a global perspective, the
Fig. 4  a Schematic diagram of sampling candidates from 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 adopt binary code to represent the offspring architectures, and the binary code (1) represents the architecture in (b)

feature values of re-parameterization operations are multiplied by the weight $\beta$ of the current block. When selecting the re-parameterization operations under resource constraints, the algorithm preferentially retains the re-parameterization operations in blocks that are important for improving the overall network performance. Therefore, when generating offspring architectures, both the global and local characteristics of the architecture are considered.

The architectures in the 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 generated according to importance are used to form the third part of the population. Specifically, the random distribution noise $\sigma_\alpha$ and $\sigma_\beta$ are added to the architecture parameters $\alpha$ and $\beta$ 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 $\alpha_0$ and $\beta_0$ are the initialization weights of $\alpha$ and $\beta$. The range of the perturbation can be defined as:

$$\sigma_\alpha \in (\alpha'_0 - \max(\alpha'), \alpha'_0 - \min(\alpha'))$$

(8)

$$\sigma_\beta \in (\beta'_0 - \max(\beta'), \beta'_0 - \min(\beta'))$$

(9)

where $\sigma_\alpha$ and $\sigma_\beta$ 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 described as:

$$\begin{cases} 
1, \text{ if rank} \left[ (\beta'_i + \sigma_\beta) \cdot (\alpha'_i + \sigma_\alpha) \right] \leq C \\
0, \text{ else}
\end{cases}$$

(10)

where 1-element means the network uses this connection. rank $(\cdot)$ denotes the global ranking. As shown in Fig. 4, the red line indicates the selected branch, and the black line indicates not selected. Due to the addition of appropriate perturbations, the candidate operations with
Algorithm 1 Directional evolution strategy for neural architecture search

Require: SuperNet \( S \), Population \( P = \{P_1, \cdots, P_k\} \), evolution number \( E_{evo} \), Warm up number \( E_{warm} \), parameter optimization epochs \( E_{p} \), arch-parameters \( \alpha, \beta \).

1: \textbf{while} \( i < E_{warm} \) \textbf{do}
2: \quad Warm up SuperNet \( S \)
3: \textbf{end while}
4: \textbf{while} \( j < E_{evo} \) \textbf{do}
5: \quad \textbf{while} \( k < E_{p} \) \textbf{do}
6: \quad \textbf{for} Mini-batch data X, target Y in Dataset \textbf{do}
7: \quad \quad Random sample B sub-architectures from Population.
8: \quad \quad Forward B sampled sub-networks.
9: \quad \quad Calculate loss and compute the gradients according to Eq. (3).
10: \quad \quad Update network parameters \( \omega \) and architecture parameters \( \alpha, \beta \).
11: \quad \textbf{end for}
12: \quad \textbf{end while}
13: \quad Re-parameterize architecture and obtain the performance of the architecture.
14: \quad Select the architecture according to performance and perform mutation crossover. Meanwhile, sample the sub-architectures from the SuperNet according to Eqs. (8)-(10).
15: \textbf{end while}
16: Output: Architectures \( P = \{P'_1, P'_2, \cdots, P'_k\} \).

high weight are retained and do not completely ignore the operations with low weight in the current stage.

3.4 Performance Estimation of Population

In this section, we proposed a method to speed up the process of obtaining architecture fitness and explains the advantages of this verification method.

In evolutionary algorithm-based NAS, evaluating the architectures takes much time. In addition, there is still a tiny deviation in the architecture performance before and after re-parameterization. Although this deviation can be ignored in practical application, we need to search for the best performance architecture after re-parameterization. Thus, in this work, the re-parameterization operations are fused into the fixed operation before verifying architecture performance, as shown in Fig. 5. 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 \( \alpha \) and \( \beta \) to indicate the importance of blocks and candidate operations. The weights and biases of the candidate operations need to be scaled \( \alpha \cdot \beta \) times before the architecture is re-parameterized. Table 2 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. Considering factors such as architecture diversity, search time, and computational resources in this experiment, the population size is set to 128. The overall training procedure is summarized in Algorithm 1.
Efficient Re-parameterization Operations Search for…

Fig. 5 Using the re-parameterized structure to obtain accuracy. Each sub-architecture in the population can be re-parameterized into the rightmost structure

Table 2 We obtain the performance of architecture on Nvidia A100 GPU

| Population size/time | 64/S | 128/S | 256/S |
|----------------------|------|-------|-------|
| Multi-Branch VGG-16 [17] | 1349.4 | 2681.1 | 5336.9 |
| Resnet-18 [18] | 649.1 | 1301.6 | 2685.7 |
| Re-parameterized VGG16 [17] | 499.5 | 1000.0 | 1823.9 |
| Resnet-18 [18] | 286.3 | 571.5 | 1133.9 |
| Acceleration percentage (%) VGG-16 [17] | 63.7 | 62.7 | 65.8 |
| Resnet-18 [18] | 55.9 | 56.1 | 57.8 |

The time to evaluate the architecture performance 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)

4 Experiments

In this section, to verify that 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 the residual structure of ResNet, we remove the residual connection from the search space to avoid the optimization difficulties caused by the diversity of gradients. Our experiment is divided into two stages: the search stage and the retraining stage.

4.1 Search Architectures on CIFAR-10

In this section, we search a set of re-parameterization operations on the CIFAR-10 dataset for ResNet-18 and VGG-16 to verify the effectiveness of the improved search space and the search algorithm.

We add a re-parameterization block similar to the residual structure to the $3 \times 3$ convolution in VGG-16 and ResNet-18. For a fair comparison, the network structure and data augmentation techniques are followed by ACNet [39], and ResNet [18]. We adopt the SGD optimizer with a learning rate of 0.1 to optimize the parameters of the network. To optimize the architecture parameters $\alpha$ and $\beta$, we use Adam optimizer with learning rate of 0.0001
Table 3 Comparison with state-of-the-art image classifiers on the CIFAR-10 dataset

| Model                  | Top-1 (%) | Params(M) | Inference (ms) | Search cost (GPU days) |
|------------------------|-----------|-----------|----------------|------------------------|
| VGG [17]               | 94.12     | 14.73     | 2.14           | –                      |
| ResNet-18 [18]         | 96.21     | 11.69     | 4.12           | –                      |
| RepVGG [28]            | 94.62     | 14.73     | 1.76           | –                      |
| RepNAS (VGG) [29]      | 95.43     | 11.69     | 1.76           | 0.7                    |
| AcNet [39]             | 94.47     | 14.73     | 1.76           | –                      |
| DARTS [1]              | 97.24     | 3.3       | 31             | 4                      |
| P-DARTS [40]           | 97.50     | 3.4       | 33             | 0.3                    |
| Distilldarts [26]      | 97.57     | 3.7       | 31             | 2.3                    |
| Shapley-NAS [35]       | 97.57     | 3.6       | 32             | 0.3                    |
| CARS [7]               | 97.38     | 3.6       | 27             | 0.4                    |
| AmoebaNet-A [41]       | 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                   |

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) and (0.5, 0.999) betas. We limit the number of branches to \( \frac{2}{3} \) times 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 first warmup SuperNet lasts for 15 epochs. Then, we search for 500 epochs and retrain architectures on the CIFAR-10 dataset. Except for the learning rate and the probability of the drop-path, the retraining process is the same as DARTS [1]. Respectively, the learning rate and drop-path probability is set to 0.05 and 0.08.

Table 3 shows our results. We achieved 1.02% better accuracy than RepVGG [28] and 0.21% than RepNAS [29]. Our architecture has a great advantage in the inference process. Since the re-parameterized architecture retains only convolution, residual operation, 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 [1], Distilldarts [26], Shapley-NAS [35], etc..

4.2 Experience on ImageNet-1K

In this section, we first search the re-parameterization operation for the ResNet-18, ResNet-34, and ResNet-50 in the ImageNet-1K dataset to verify the generalization ability of the re-parameterization architectures, and then analyze the structure and performance of the searched re-parameterization network.

4.2.1 Hyper-parameters Setting on ImageNet-1K

We evaluate architecture on the ImageNet-1K to reveal the generalization ability, which contains 1.3M images for training and 50K for validation from 1000 classes. To save computational resources and speed up the search, based on the conclusion of ACNet [39], we remove the \( 2 \times 2 \) 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
Table 4 Results of our models on ImageNet-1K dataset compared to other NAS methods and models. All experiments on the ImageNet-1K were performed based on Nvidia A100 GPU

| Model           | Top-1(%) | Top-5(%) | Params(M) | Inference (ms) | Search cost (GPU-days) |
|-----------------|----------|----------|-----------|----------------|------------------------|
| ResNet-18 [18]  | 69.76    | 89.07    | 11.69     | 4.25           | –                      |
| ResNet-34 [18]  | 73.31    | 91.42    | 21.80     | 6.12           | –                      |
| ResNet-50 [18]  | 76.10    | 93.29    | 25.56     | 7.54           | –                      |
| DyRep [30] (ResNet-18) | 71.58 | –       | 16.90     | 3.59           | –                      |
| DyRep [30] (ResNet-34) | 74.68 | –       | 33.10     | 5.15           | –                      |
| DyRep [30] (ResNet-50) | 77.08 | –       | 31.50     | 6.14           | –                      |
| DDB [27] (ResNet-18) | 70.99 | –       | 26.30     | 3.59           | –                      |
| DDB [27] (ResNet-34) | 74.33 | –       | 49.90     | 5.15           | –                      |
| DDB [27] (ResNet-50) | 76.71 | –       | 40.70     | 6.14           | –                      |
| DARTS [1] (second) | 73.30 | 91.3    | 4.70      | 67.4           | 4                      |
| P-DARTS [40]    | 75.60    | 92.6     | 4.90      | 62.3           | 0.3                    |
| Distilldarts [26] | 75.80 | –       | 5.50      | 57.1           | 0.5                    |
| Shapley-NAS [35]| 76.10    | –       | 5.40      | 59.7           | 4.2                    |
| CARS [7]        | 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                     |

We calculated the parameters of the model and tested the model of the inference time with a batch size of 1, full precision (fp32).

optimize $\alpha$, $\beta$. We limit the number of branches to $\frac{1}{2}$ times of the total branch number. The SuperNet warmup stage lasts for 15 epochs and we search for 200 epochs, then fix the structure of the SuperNet and retrain 120 epochs. To be fair, we adopt the same data augmentation techniques as ResNet [18].

We compare our architectures with state-of-the-arts in Table 4. 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 [30] and 1.21% higher than DDB [27]. IrepResNet-34 and IrepResNet-18 also achieve great performance. Meanwhile, the architecture has faster inference speed than ResNet, DARTS [1], P-DARTS [40], Distilldarts [26], Shapley-NAS [35], etc. However, the population is iterated on the dataset several times in each epoch. After that, the optimal architecture of the population needs to be retrained. So more computing resources are consumed compared with similar work [29, 30].

4.2.2 Analysis of IrepResNet Architecture

We plot the structure of IrepResNet in Appendix A.1. 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. 6, it can be easily concluded that IrepResNet-50 has stronger discrimination ability for targets compared to ResNet-50.
Fig. 6 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 Conv1x1-Conv3x3-Conv1x1 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.

In the general visual frameworks, bottom layers play more roles in capturing local information (local color features, texture features and shape features, etc.). While top layers play more in modeling global information. For ResNet-50, the role of the first eight layers is mainly to achieve the separation of foreground and background in the image, because it can focus on more detailed information to locate the position of the target in the picture. The last eight layers have a large field of perception, their task is mainly further to distinguish target and background from global perspective. This division of task is significant for the formation of the IrepResNet-50 structure.

The first eight layers are close to the input, for neural networks, being close to the input means that the network can not directly obtain high semantic information from features, making it challenging to process features. In addition, due to the shallow layers having few channels, their ability to obtain features is poor.

Thus, the re-parameterization operations are essential to improve the current feature information, making them a higher weight. After the feature extraction of the shallow layers, the last eight layers can obtain the input features with high semantic information. And the last eight layers can acquire more feature information because of the more channels, which allows them to take on the task of focusing on the target and coarse-grained adjustment of the target and background in the picture. Compared with the first eight re-parameterization blocks, the re-parameterization operations is not important for the last eight layers, which makes the re-parameterization operations have a smaller weight. Therefore, searching the architecture under resource constraints makes the algorithm prefer retaining the re-parameterization operations in the shallow layers, which leads to the searched re-parameterization architecture to be truncated from the middle architecture. 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.
4.2.3 Reasons for Achieving Good Results

We analyzed the reasons that the performance of IrepResNet exceeded the relevant work, which can be summarized as follows: (1) Design and selection of the re-parameterization operations. Due to the cross-shaped weight at the center position of square convolution contains the most of feature information. We focus on enhancing the feature information of the position when designing the re-parameterization operations. Therefore, we can enhance the convolution network with limited resources to achieve maximum performance improvement. (2) Expanded search space. In Sect. 3.1, we show that our search space is $8^N$ times that of RepNAS [29] and DyRep [30]. As far as ResNet-50 is concerned, $N = 16$, the search space of IrepResNet-50 expanded by $2.81 \times 10^{14}$ times, which supports us in searching for a better architecture. (3) Directed search algorithm. The proposed directed evolution algorithm is a greedy search algorithm, which aims to find a better architecture when the architecture is optimal. In order to overcome the randomness of the evolutionary algorithm in generating new individuals, we introduce the learnable architecture parameters $\alpha$ and $\beta$, which describe the importance of candidate operations and blocks. In each new iteration, we use the architecture parameters $\alpha$ and $\beta$ to balance the global and local importance of the operations to generate different subnets (see Sect. 3.3), which accelerates the convergence of the algorithm. Although the introduction of architecture parameters may lead to the Matthew effect in the optimization process, the mutation and crossover encourage the algorithm to explore more architectures and effectively alleviate the Matthew effect. With enough iterations, the algorithm will search for an excellent architecture from the search space.

4.3 Generalization Performance on Downstream Task

In this section, we deploy the re-parameterization architecture to the downstream tasks to verify that our architecture also performs well in the downstream tasks.

We transfer our ImageNet-pretrained IrepResNet-50 and IrepResNet-18 models to downstream tasks object detection. Specifically, the pre-trained model is used as the backbone for the downstream algorithms FPN [42] and CenterNet [43] algorithms on the COCO dataset [44]. For the optimization of the target detection model, we refer to the optimization approach and hyperparameter settings of MMDetection [45]. FPN and CenterNet 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 5 show that IrepResNet can achieve better performance compared to FPN, CenterNet, and DyRep.

| Backbone               | Algorithm | ImageNet Top-1 | COCO mAP |
|------------------------|-----------|----------------|----------|
| ResNet-18 [18]         | CenterNet | 69.76          | 29.5     |
| ResNet-50 [18]         | FPN       | 76.10          | 37.9     |
| DyRep(ResNet-50) [30]  | FPN       | 77.08          | 38.1     |
| IrepResNet-18          | CenterNet | 71.27          | 31.2     |
| IrepResNet-50          | FPN       | 77.92          | 38.2     |
Table 6 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 | 95.52 |

Fig. 7 Differential heatmap of the $1 \times 3$ convolution–$1 \times 2$ dilated convolution and $3 \times 1$ convolution–$2 \times 1$ dilated convolution, which searched on CIFAR-10 and ImageNet-1K dataset.
4.4 Ablation Study

In this section, we first analyze the impact of different resource constraints and re-parameterization operations on the performance of the re-parameterization architecture. Then, we explored the effect of search time on architecture performance.

4.4.1 Searching Under Different Resource Restrictions

To explore the impact on the architecture performance under different resource constraints, we searched IrepVGG-16 and IrepResNet-18 on the CIFAR-10 dataset. Specifically, the number of branches is set to \([\frac{1}{6}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{5}{6}, 1]\) times of the total branch number. As shown in Table 6, when the resource constraint reaches 2/3, the architecture achieves better performance. We found that the architecture performance is weaker than RepNAS [29] when retaining the same number of branches as RepNAS in the improved re-parameterization search space (retain four branches for each block). Based on AcNet [39], 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. 7, 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 a 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) is stronger than dilated convolutions in this experiment. Therefore, when the same number of branches are retained as RepNAS [29], 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 [29].

4.4.2 The Impact of Search Time on Architecture Performance

For NAS, the architecture performance depends on the search time. Generally, the longer the search time, the better the architecture performance. In order to explore whether the proposed algorithm can find a better re-parameterized architecture with the increase in search time. We
Fig. 9 IrepResnet-50 searched on ImageNet. The $3 \times 3$ convolution operation is a fixed operation and does not participate in the search process of the architecture.
Efficient Re-parameterization Operations Search for…

Fig. 10 List of architectures of IrepResNet-50, IrepResNet-18, IrepResNet-34. In the re-parameterization search space, the order of candidate operations are $[1 \times 1, \text{AVG}, 1 \times 1 - 3 \times 3, 1 \times 3, 3 \times 1, D i - 1 \times 2, D i - 2 \times 1]$. The 0–1 element of each line in the list corresponds to the candidate operations one by one. Each row represents the enhancement of a $3 \times 3$ convolution. From left to right, it represents the re-parameterization structure from the first layer to the last layer.

Fig. 11 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 Conv$3 \times 3$-Conv$3 \times 3$ as one layer.

output the current optimal architecture every 50 epochs and retrain it. Specifically, we did experiment on the CIFAR-10 dataset, and the settings of the hyper-parameters are the same as those in Sect. 4.1. As shown in Fig. 8, we plot the convergence curve of the architecture accuracy in the search process. As the number of epochs increases, the performance of the best architecture and the population increases gradually and tends to be stable. By retraining the best architecture, we verified that the proposed algorithm can search for a better architecture with increased search time. However, when the search time reaches a certain epochs (about 400 epochs), the architecture performance will no longer increase significantly.
5 Conclusion

In order to further improve traditional convolutional networks, we designed a larger reparameterization search space and added it as a re-parameterization block to each $3 \times 3$ convolution in the ResNet model. Then the proposed directional evolutionary strategy is used to search for the optimal architecture from these re-parameterization search space. At the same time, we solved the optimization problem caused by residual operation when searching for re-parameterization blocks for ResNet-style networks and explained the reason for the formation of the final architecture. Extensive experiments demonstrate that the proposed improved re-parameterization search space can further improve the performance of models and perform well in downstream tasks. Compared with similar work, we find a better set of re-parameterization architecture and achieve better performance. It is worth mentioning that our search space can be used as a bridge between coarse-grained search and fine-grained search compared with the previous re-parameterization search space. It means that the re-parameterization model after coarse-grained search (architecture operation) can be divided into $1 \times 1$ convolution and $2 \times 2$, $2 \times 1$, $1 \times 2$ dilated convolution. Then the model can be unstructured pruned, which can further reduce the FLOPs of the model. In the future, we will design an algorithm suitable for unstructured pruning.

Author Contributions Wang Xiaowei wrote manuscript text, did experiments and prepared figures. All authors reviewed the manuscript.

Declarations

Conflict of interest The authors declare no conflict of interest.

Appendix A

A.1 IrepResNet Model

A.2 Proof of Gradient Diversity

For the re-parameterization block, all parallel convolution operations can be fused into square convolution, which can be written as:

$$f(w) = f_{k \times k} + \text{Rep} \left( \sum_{o \in O} f_o (w_o) \right)$$

(A1)

where $f(\cdot)$ represents convolution operation and $w$ represents the learnable parameters of the convolution operation. $O$ is the operation in the improved search space. When the residual connection is removed from the re-parameterization search space, the output feature of layer $L$ can be expressed as:

$$x_L = x_l + \sum_{i=l}^{L-1} f \left( f \left( x_i, w_{2i-1} \right), w_{2i} \right)$$

(A2)

where $x_l$ is the output feature of $l_{th}$ residual unit ($f(w_1)$ and $f(w_2)$ compose a residual unit) and $l < L$. $f \left( x_i, w_{2i-1} \right)$ is the output of the first operation in the residual unit, such as $f \left( w_1 \right)$ and $f \left( w_3 \right)$.
When the residual operation is not removed, from Fig. 12a, we can get the follow output feature:

\[ x_L = x_l + \sum_{i=1}^{L-1} f( f(x_i, w_{2i-1}) + x_i, w_{2i}) + f(x_l, w_{2i-1}) + x_i \]  

(A3)

where \( f(x_i, w_{2i-1}) + x_i \) can be regarded as the input of the second operation (\( f(w_2) \) and \( f(w_4) \)) in the residual unit or the sum of the first operation output and its corresponding residual connection output.

We assume that the loss function of the network is \( L \). According to the chain derivative rule, the following formulas is obtained from Eq. (A2) and Eq. (A3):

\[
\frac{\partial L}{\partial x_l} = \frac{\partial L}{\partial x_L} \cdot \frac{\partial x_L}{\partial x_l} = \frac{\partial L}{\partial x_L} \cdot \left( 1 + \frac{\partial \sum_{i=1}^{L-1} f( f(x_i, w_{2i-1}), w_{2i})}{\partial x_l} \right) \\
\frac{\partial L}{\partial x_L} = \frac{\partial L}{\partial x_L} \cdot \frac{\partial x_L}{\partial x_l} = \frac{\partial L}{\partial x_L} \cdot \left( 1 + \frac{\partial \sum_{i=1}^{L-1} f( f(x_i, w_{2i-1}) + x_i, w_{2i})}{\partial x_l} \right) \\
+ \frac{\partial L}{\partial x_l} \cdot \left( \frac{\partial \sum_{i=1}^{L-1} f( x_i, w_{2i-1})}{\partial x_l} \right) + \frac{\partial L}{\partial x_L} \cdot \left( \frac{\partial \sum_{i=1}^{L-1} x_i}{\partial x_l} \right)
\]  

(A4)

It is found that when taking the derivative of the input variable, Eq. (A4) uses the feature information of residual units in layer \( l \rightarrow L \), Eq. (A5) uses the output feature information of residual units in layers \( l \rightarrow L, l \rightarrow (2i-1)(i = l, l+1...L) \) and \( l \rightarrow (L-1) \). Therefore, Eq. (A4) only considers the unique gradient information when updating the parameters. While, Eq. (A5) needs to consider gradient information from three different layers and use
them to update the same parameters to make it optimal, which brings challenges to parameter optimization. At the same time, since the gradient has a direction, this inevitably makes the descent trajectory of Eq. (A5) different from Eq. (A4). Therefore the diversity of gradient information destroys the residual structure of the original ResNet, which makes the network more difficult to optimize than the original network.

References

1. Liu H, Simonyan K, Yang Y (2019) Darts: differentiable architecture search. In: International conference on learning representations
2. Hu S, Xie S, Zheng H, Liu C, Shi J, Liu X, Lin D (2020) Dsnas: direct neural architecture search without parameter retraining. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 12084–12092
3. Wang H, Yang R, Huang D, Wang Y (2021) Idarts: improving darts by node normalization and decorrelation discretization. IEEE Transactions on Neural Networks and Learning Systems
4. Zhou Q, Zheng X, Cao L, Zhong B, Xi T, Zhang G, Ding E, Xu M, Ji R (2021) Ec-darts: inducing equalized and consistent optimization into darts. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 11986–11995
5. Lu Z, Whalen I, Boddeti V, Dhebar Y, Deb K, Goodman E, Banzhaf W (2019) Nsga-net: neural architecture search using multi-objective genetic algorithm. In: Proceedings of the genetic and evolutionary computation conference, pp 419–427
6. Chu X, Zhou T, Zhang B, Li J (2020) Fair darts: eliminating unfair advantages in differentiable architecture search. In: European conference on computer vision, Springer, pp 465–480
7. Yang Z, Wang Y, Chen X, Shi B, Xu C, Xu C, Tian Q, Xu C (2020) Cars: continuous evolution for efficient neural architecture search. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 1829–1838
8. Guo Z, Zhang X, Mu H, Heng W, Liu Z, Wei Y, Sun J (2020) Single path one-shot neural architecture search with uniform sampling. In: European conference on computer vision, Springer, pp 544–560
9. Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 779–788
10. Redmon J, Farhadi A (2018) Yolov3: an incremental improvement. Preprint arXiv:1804.02767
11. Liu C, Chen L-C, Schroff F, Adam H, Hua W, Yuille AL, Fei-Fei L (2019) 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
12. Fang J, Sun Y, Zhang Q, Peng K, Li Y, Liu W, Wang X (2020) Fna++: fast network adaptation via parameter remapping and architecture search. IEEE Trans Pattern Anal Mach Intell 43(9):2990–3004
13. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc. IEEE 86(11):2278–2324
14. Zhu J-Y, Park T, Isola P, Efros AA (2017) Unpaired image-to-image translation using cycle-consistent adversarial networks. In: Proceedings of the IEEE international conference on computer vision, pp 2223–2232
15. Weng L, Wang Y, Gao F (2022) Traffic scene perception based on joint object detection and semantic segmentation. Neural Process Lett 1–17
16. Xiao T, Li S, Wang B, Lin L, Wang X (2017) Joint detection and identification feature learning for person search. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 3415–3424
17. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. In: International conference on learning representations
18. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
19. Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H (2017) Mobilenets: efficient convolutional neural networks for mobile vision applications. Preprint arXiv:1704.04861
20. Sandler M, Howard A, Zhu M, Zhmoginov A, Chen L-C (2018) Mobilenetv2: inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4510–4520
21. Howard A, Sandler M, Chu G, Chen L-C, Chen B, Tan M, Wang W, Zhu Y, Pang R, Vasudevan V et al. (2019) Searching for mobilenetv3. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 1314–1324
22. Huang G, Liu Z, Van Der Maaten L, Weinberger KQ (2017) Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4700–4708
23. Wan A, Dai X, Zhang P, He Z, Tian Y, Xie S, Wu B, Yu M, Xu T, Chen K et al. (2020) 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
24. Hong W, Li G, Zhang W, Tang R, Wang Y, Li Z, Yu Y (2020) Dropnas: Grouped operation dropout for differentiable architecture search. In: International joint conference on artificial intelligence (IJCAI), pp 2326–2332
25. Xu Y, Xie L, Zhang X, Chen X, Qi G-J, Tian Q, H. (2020) Pdarts: partial channel connections for memory-efficient architecture search. In: 8th international conference on learning representations
26. Liao W, Zhang Z, Wang X, Xiong H (2020) Pc-darts: partial channel connections for memory-efficient architecture search. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 588–597
27. Zhang M, You X, Rong J, Ou L, Gao F (2020) Repnas: searching for efficient re-parameterizing blocks. Preprint arXiv:2109.03508
28. Huang T, You S, Zhang B, Du Y, Wang F, Qian C, Xu C (2022) Dyrep: bootstrapping training with dynamic re-parameterization. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 10734–10742
29. Yang Y, You S, Li H, Wang F, Qian C, Lin Z (2021) 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
30. Lin T-Y, Dollár P, Girshick R, He K, Hariharan B, Belongie S (2017) Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2117–2125
31. Zhou X, Wang D, Krähenbühl P (2019) Objects as points. Preprint arXiv:1904.07850
44. Lin T-Y, Maire M, Belongie S, Hays J, Perona P, Ramanan D, Dollár P, Zitnick CL (2014) Microsoft coco: common objects in context. In: European conference on computer vision, Springer, pp 740–755
45. Chen K, Wang J, Pang J, Cao Y, Xiong Y, Li X, Sun S, Feng W, Liu Z, Xu J et al (2019) Mmdetection: open mmlab detection toolbox and benchmark. arxiv 2019. Preprint arXiv:1906.07155

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.