Operation-Level Early Stopping for Robustifying Differentiable NAS

Anonymous

Abstract

Differentiable NAS (DARTS) is a simple and efficient automatic neural architecture search method and has been widely used in a variety of machine learning tasks. However, DARTS still has several robustness problems, mainly the domination of skip connection. The resulting architecture will be full of parametric-free operations, leading to performance collapse. Existing methods believe that the skip connection has additional advantages in optimization compared to other parametric operations, and propose to alleviate the domination of skip connection by eliminating these additional advantages. In this paper, we analyze this issue from a simple and straightforward perspective, proposing that the domination of skip connection is due to the fact that parametric operations overfit the training data and gradually deviate from the validation data while architecture parameters are trained on the validation data. Based on this concept, we propose the operation-level early stopping (OLES) method to solve this problem and robustify DARTS without introducing any computation overhead. Extensive experimental results can verify our conjecture and the effectiveness of our method.

1 Introduction

Neural architecture search (NAS) aims to automatically design high-performance neural architectures [Zoph and Le, 2016] for given tasks in a data-driven manner. Deep models designed by NAS have achieved state-of-the-art performance on a variety of tasks, such as image classification [Termritthikun *et al.*, 2021; Real *et al.*, 2018; Pham *et al.*, 2018] and semantic segmentation[Liu *et al.*, 2019]. Among various NAS methods, differentiable architecture search (DARTS) [Liu *et al.*, 2018] significantly improves the efficiency of NAS by parameter-sharing and continuous relaxation. It converts the discrete architecture selections into continuous architecture parameters α and optimizes them via gradient-based optimizers.

Despite its simplicity and efficiency, DARTS still has several issues that make it difficult to outperform other NAS methods, even a simple random search [Li and Talwalkar, 2020; Yu *et al.*, 2019], in many cases. The first issue is the *performance gap*. Due to the discrepancies between derived discrete architectures and continuous architecture parameters, the derived model shows non-negligible performance degradation [Liu *et al.*, 2018]. In addition, DARTS also suffers from serious *robustness problems* [Nayman *et al.*, 2019; Chen *et al.*, 2019; Liang *et al.*, 2019; Chu *et al.*, 2020; Zela *et al.*, 2019; Wang *et al.*, 2021]. In particular, in the search phase of DARTS, when the number of training epochs increases, the searched architectures will fill with parametric-free operations [Chu *et al.*, 2019] such as skip connection or even random noise, leading to performance collapse. These issues waste a lot of computational resources and seriously hinder the application of DARTS.

The domination of skip connection is the main robustness problem of DARTS [Wang *et al.*, 2021]. Many studies have been conducted to explore why the domination of skip connection happens and how to alleviate this problem. Most of them believe that the skip connection exhibits unfair advantages in the gradient-based optimization of the search phase [Liang *et al.*, 2019; Chu and Zhang, 2020; Chu *et al.*, 2019] or plays an additional role in stabilizing the supernet training by constructing a residual block [Chu *et al.*, 2020]. They propose to alleviate the domination of skip connection by eliminating these unfair advantages [Chu *et al.*, 2019] or separating the role [Chu *et al.*, 2020]. These ideas mainly come from ResNet [He *et al.*, 2015]. However, the generic optimization in ResNet is fundamentally different from the bi-level optimization in DARTS. Observations and theories based on the uni-level optimization are not enough to support the claim that the skip connection still has these unfair advantages in DARTS. The work [Wang *et al.*, 2021] analyzes this problem from the perspective of architecture selection, proposing that the magnitude of the architecture parameter may not reflect the strength of the operation. And based on this assumption, a perturbation-based architecture selection method is proposed. This idea, however, conflicts with the fundamental motivation of DARTS, which utilizes the continuous relaxation to approximately solve the combinatorial optimization problem.

Different from the above sophisticated assumptions, in this paper we consider the reasons for the domination of skip connection from a simpler and more straightforward perspective. We observe that the robustness problem of DARTS is due to overfitting. Specifically, the parameters of operations (e.g. convolution) in the supernet overfit the training data and gradually deviate from the validation data, while the architecture parameters are trained on the validation data, so the advantages of parametric operations are increasingly weakened and finally parametric-free operations dominate. It is obvious that the supernet is overparameterized, and the architecture parameters also affect the supernet training. Therefor, operations that show advantages in the early stage of search are more prone to overfitting.

Based on the above observations, to solve the domination of skip connection problem, we further propose an operation-level early stopping (OLES) method that monitors whether each operation in the supernet tends to overfit the training data and stops training the operation when it overfits. Specifically, we utilize the difference between the gradient directions of operation parameters on the training data and the validation data to verify whether the operation tends to overfit. If the difference keeps being large in multiple consecutive iterations, the operation will stop training. By this means, the domination of skip connection problem can be alleviated. And the distribution of architecture parameters will become sharper with more search iterations, which can also reduce the performance gap caused by discretization. Moreover, our method only needs to maintain the directions of operation gradients, and the additional overhead can be negligible.

In summary, we make the following contributions:

- We analyze the robustness problem of DARTS from a new perspective of overfitting of operations in the supernet and conduct motivational experiments to verify our conjecture.
- We propose the operation-level early stopping method to solve the domination of skip connection in DARTS with negligible computation overheads.
- Extensive experiments on the DARTS' search space and NAS-Bench-201 [Dong and Yang, 2020] demonstrate the effectiveness and efficiency of our method, which can achieve state-of-the-art (SOTA) performance on multiple commonly used image classification datasets. Specifically, we achieve new SOTA test errors of 2.29$\%$ and 17.10$\%$ on CIFAR-10 and CIFAR-100, respectively.

2 Background and Related Work

2.1 Neural Architecture Search

Over the years, a variety of NAS methods, including evolutionary-based [Termritthikun *et al.*, 2021; Real *et al.*, 2018; Guo *et al.*, 2020], reinforcement-learning-based [Zoph and Le, 2016; Pham *et al.*, 2018], and gradient-based [Liu *et al.*, 2018; Xie *et al.*, 2018; Cai *et al.*, 2018] methods, have been proposed and have achieved great success in many computer vision (CV) and natural language processing (NLP) [So *et al.*, 2019] tasks. As a pioneering work, NAS [Zoph and Le, 2016] uses an RNN controller to generate architectures and employs the policy gradient algorithm to train the controller. However, this method suffers from low search efficiency. To improve the search efficiency, the cell-based micro search space [Zoph *et al.*, 2017] and weight-sharing mechanism [Pham *et al.*, 2018] have been proposed. DARTS [Liu *et al.*, 2018], which also takes advantage of weight-sharing, utilizes the gradient-based method to optimize architecture parameters and model weights alternatively.

2.2 Preliminaries of Differentiable Architecture Search

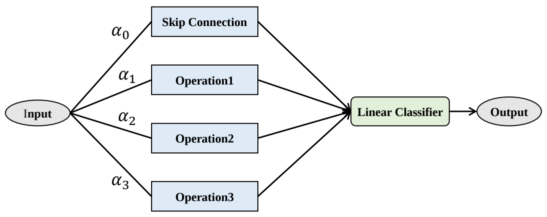
We will now review DARTS in more detail. The original DARTS relies on cell-based micro search space. Each cell contains N nodes and E edges that are organized in a directed acyclic graph (DAG). And each node represents a feature map $x^{(i)}$, and each edge is associated with an operation $o \in \mathcal{O}$, where $\mathcal{O}$ denotes the set of candidate operations (e.g., skip connect, sep conv 3x3, etc.). During architecture search, instead of applying a single operation to a specific node, continuous relaxation is applied to relax the categorical choice of a specific operation to a mixture of candidate operations, i.e., $\bar{o}^{(i,j)}(x) = \sum_{o \in \mathcal{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^{(i,j)})} o(x)$. $\alpha = \{\alpha^{(i,j)}\}$ serves as the architecture parameters. DARTS then jointly optimizes architecture parameters α and model weights w with the following bi-level objective via alternative gradient descent:

$$\begin{aligned} &\min_{\alpha} \mathcal{L}_{val}(w^*, \alpha) \\ \text{s.t.} \quad & w^* = \arg\min_{w} \mathcal{L}_{train}(w, \alpha). \end{aligned}$$

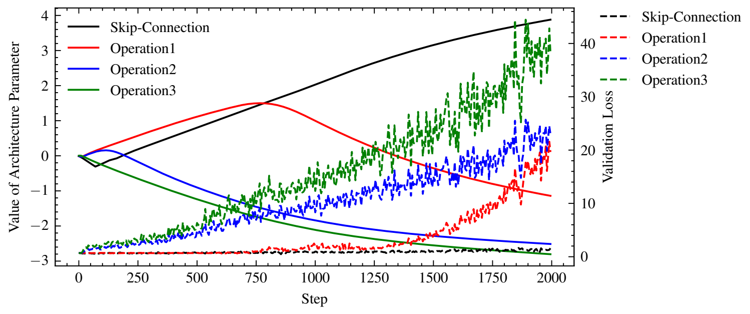
Since DARTS follows the weight-sharing mechanism, we refer to the continuous relaxed network as the "supernet". At the end of the search phase, a discretization phase is performed by selecting operations with the largest α values to form the final architecture.

2.3 Robustifying DARTS

Despite its simplicity and efficiency, the undesired domination of skip connection is often observed in DARTS. Therefore, endeavors have been made to remedy the performance collapse of DARTS caused by the domination of skip connection. Prior works mostly emphasize eliminating the domination of skip connections through diminishing their unfair advantages. DARTS+[Liang *et al.*, 2019] directly uses a fixed value to limit the number of skip connections in each cell. In addition to adding a hard constraint to the number of skip connections, Progressive DARTS[Chen *et al.*, 2019] further introduces the operation-level Dropout. DARTS$-$[Chu *et al.*, 2020] introduces an auxiliary skip connection to separate the unfair advantage of the skip connection by analyzing the skip connection from the perspective of ResNet. Another strategy is to remove the exclusive competition of operation selection in DARTS. FairDARTS[Chu *et al.*, 2019] utilizes *Sigmoid* to replace the *Softmax* relaxation of DARTS and offers an independent architecture parameter for each operation. Furthermore, NoisyDARTS[Chu and Zhang, 2020] injects unbiased noise into the candidate operations to ensure that the good ones win robustly. However, DARTS+PT [Wang *et al.*, 2021] believes that the magnitude of the architecture parameter is not a good indicator of operation strength and proposes a perturbation-based architecture selection method.



(a) The search space of motivational experiment. Operations here are MLPs with different numbers of parameters, and α denotes the architecture parameter.



(b) The architecture parameter and validation loss of each operation. The solid line represents the value of architecture parameter, and the dotted line represents the validation loss.

Figure 1: The motivational experiment. As the number of training steps increases, the validation loss of the parametric operation increases significantly, while the corresponding architecture parameter gradually decreases.

3 Motivation and Methodology

In this section, we first demonstrate that the domination of skip connection is due to the overfitting of operations in the supernet through a simple motivational experiment. Then, based on our observations and motivation, we propose an operation-level early stopping (OLES) method to solve this issue.

3.1 The Domination of Skip-Connection is Due to Overfitting

In this paper, we argue that the reason for the domination of skip connection is that operations in the supernet overfit the training data, such that these operations deviate increasingly from the distribution of the validation data. However, the architecture parameters are trained on the validation data, essentially selecting operations based on the validation performance. Therefore, as the number of DARTS training iterations increases, the parametric operations overfit more and more seriously, and the parametric-free operations gradually dominate. In this subsection, we present empirical evidence on the relation between the overfitting of operations in the supernet and architecture parameters.

We first generate a simple binary classification dataset by taking points from two classes on the plan according to a classification boundary. The dataset is very prone to overfitting, which can help us better verify our idea. Then, as shown in Figure 1a, we construct a search space (supernet) which only contains one set of candidate operations, and the DARTS is applied to select one operation from them. The candidate operations contain a skip connection and a series of multi-layer perceptrons (MLPs) with different numbers of parameters. It is obvious that the network with more parameters will overfit our dataset more seriously. During the search phase of DARTS, we record the architecture parameter and the validation loss with the corresponding output of each operation. The supernet and architecture parameters are trained using standard DARTS with the mixture output of all the operations; however, the validation loss of each operation is calculated directly with its output. We ignore other losses for simplicity. As shown in Figure 1b, we can observe that the validation losses of all parametric operations start to increase after a few iterations, and the architecture parameter exhibits an obviously negative correlation with the validation loss. The architecture parameter corresponding to the skip connection decreases in the early stage of training, then gradually increases, and finally dominates.

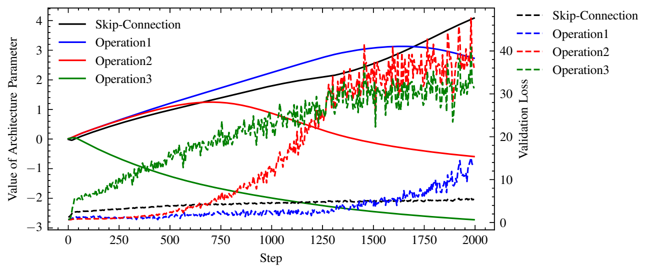


Figure 2: The architecture parameter and validation loss of each operation in FairDARTS.

Because most existing works believe that the domination of skip connection is due to the unfair advantage of the skip connection. Following the settings of FairDARTS [Chu *et al.*, 2019], we replace the *Softmax* relaxation of the above experiment with the *Sigmoid* relaxation. The unfair advantage of the skip connection will be eliminated by removing the exclusive competition among operations. Then we rerun the experiment, and the results are shown in Figure 2. We can observe that, after removing the exclusive competition, the architecture parameters still negatively correlate with the validation losses, similar to that in Figure 1b. It illustrates that the unfair advantage is not the fundamental reason for the domination of skip connection, but rather the effect of overfitting of operations in the supernet on the training of architecture parameters.

Although this experiment is simple and exaggerates the overfitting phenomenon, it is enough to demonstrate the essential reason for the domination of skip connection. In real-world applications of DARTS, since the supernet is always overparameterized and the training data for the supernet is

Algorithm 1 DARTS with operation-level early stopping

```
Require: Supernet weights $w$; Architecture parameters $\alpha$;
    Number of search iterations $I$;
    Threshold of difference for verifying overfitting $\xi$;
    Patience of early stopping $K$;
Ensure: Searched architecture parameters $\alpha$.
 1: Construct a supernet
 2: for each $i \in [1, I]$ do
 3:     Get a batch of training data $B_{train}$
 4:     Update weights $w$ by $\nabla_w \mathcal{L}_{train}(w, \alpha, B_{train})$
 5:     Record the training gradient of each operation $g^o_{train}$
 6:     Get a batch of validation data $B_{val}$
 7:     Update parameters $\alpha$ by $\nabla_\alpha \mathcal{L}_{val}(w, \alpha, B_{val})$
 8:     Record the validation gradient of each operation $g^o_{val}$
 9:     Verify overfitting by $D_o > \xi$
10:     Stop the gradient of an operation if $D_o > \xi$ for consecutive $K$ iterations
11: end for
12: Derive the final architecture based on learned $\alpha$.
```

usually smaller (a validation dataset usually needs to be separated from the training dataset to train the architecture parameters), operations in the supernet will be more prone to overfitting.

3.2 Operation-Level Early Stopping

Based on the above observations and motivation, we propose an operation-level early stopping (OLES) method to alleviate the domination of skip connection problem and make DARTS more robust. During the search phase, since architecture parameters also influence the supernet training, different operations will be differently trained. Operations that shown advantages at the early stage of search will get more attention and are more prone to overfit. Furthermore, the operation is the basic unit of choice in DARTS. DARTS will select operations to form the final architecture. As a result, we perform early stopping on the operation. When the operation tends to overfit, if early stopping can be performed for each operation (i.e., the training of each single operation can be stopped during the supernet training), the overfitting problem can affect the training of architecture parameters less, meanwhile, the supernet can be fully trained.

Overfitting usually occurs when the model is so complex that it overly fits the distribution of training data but performs poorly on the validation data. When a model is overfitting, the training loss continues to decrease while the validation loss increases. At this time, the optimization direction (i.e., the direction of gradients) of the model parameters on the training data will be inconsistent with that on the validation data. Therefore, we propose to utilize the gradient direction of the parameters of an operation to judge whether it is overfitting. If the gradients on the training data and the validation data differ greatly in direction, we consider that the operation tends to overfit the training data.

Concretely, note that the search process of DARTS alternately trains the supernet on the training dataset and trains architecture parameters on the validation dataset. For an operation o, let P_o denote the parameters of o, and g^o_{train} and g^o_{val} denote the gradients of P_o on a training batch and a validation batch, respectively. We define the difference between the directions of gradients on the training data and validation data as:

$$D_o = \frac{\|g^o_{train} \odot g^o_{val} < 0\|_1}{C(P_o)}, \quad \text{where} \quad g^o_{train} = \frac{\partial \mathcal{L}_{train}}{\partial P_o}, g^o_{val} = \frac{\partial \mathcal{L}_{val}}{\partial P_o}, \tag{1}$$

where $\odot$ denotes the element-wise product, and $C(P_o)$ counts the number of parameters. When D_o exceeds a predefined threshold $\xi \in (0, 1)$, the operation is considered to tend to overfit. If $D_o > \xi$ for consecutive K iterations, we suppose that the operation o has overfitted the training data. The parameters of the operation will not be updated in the subsequent supernet training. The remaining operations and architecture parameters are trained as usual.

Through the operation-level early stopping method, the overfitting of operations in the supernet can be alleviated. Taking advantage of the fact that DARTS trains alternately on the training dataset and validation dataset, in the implementation, we only need to maintain the gradient directions of each operation on a training data batch and a validation data batch in one iteration, without introducing any additional computational overhead. Consequently, OLES can alleviate the domination of skip connection to make DARTS more robust, and we can train DARTS with more iterations to get better architectures and sharper architecture parameters for better performance. The overall algorithm of DARTS with operation-level early stopping can be found in Algorithm 1.

4 Experiments

4.1 Search Spaces and Experimental Settings

To verify the effectiveness of our method (OLES), we conduct experiments on the standard DARTS' search space and NAS-Bench-201, respectively. In both search spaces, we keep all the experimental settings the same as the original since our proposed method only needs to freeze the parameters of operation in the supernet training. In addition to the standard hyperparameters in DARTS, our method introduces two important hyperparameters, the overfitting threshold ξ and the patience K. Following Algorithm 1, the operation tends to overfit when the difference between gradient directions D_o is higher than the overfitting threshold ξ. When the operation tends to overfit for K consecutive iterations, it will stop training during the supernet training. We tune the overfitting threshold ξ and patience K for given datasets and search spaces in the following experiments. We implement our method on the basis of the open-source code of DARTS [1] and NAS-Bench-201 [2].

4.2 Performance on DARTS' CNN Search Space

CIFAR-10

Following the settings of DARTS [Liu *et al.*, 2018], we apply the *first-order* optimization, and the comparison results

[1]https://github.com/quark0/darts

[2]https://github.com/D-X-Y/NAS-Bench-201

Architecture	Test Err(%)		Params (M)	Search Cost (GPU-days)	Search Method
	CIFAR-10	CIFAR-100			
DenseNet-BC [Huang *et al.*, 2016]	3.46	-	25.6	-	Manual
ResNet + CutOut[He *et al.*, 2015]	4.61	17.8	1.7	-	Manual
NASNet-A + CutOut [Zoph *et al.*, 2017]	2.65	-	3.3	1800	RL
AmoebaNet-A [Real *et al.*, 2018]	3.34 ± 0.06	-	3.2	3150	Evolution
AmoebaNet-B [Real *et al.*, 2018]	2.55 ± 0.05	-	2.8	3150	Evolution
PNAS [Liu *et al.*, 2017]	3.41 ± 0.09	-	3.2	225	MDL
ENAS + CutOut [Pham *et al.*, 2018]	2.89	-	4.6	0.5	RL
DARTS(1st) + CutOut[Liu *et al.*, 2018]	3.00 ± 0.14	$17.76^{\diamond}$	3.3	0.4	Gradient
DARTS(2nd) + CutOut[Liu *et al.*, 2018]	$2.85 \pm 0.08^{*}$	$17.54^{\diamond}$	3.4	0.4	Gradient
SNAS[Dong and Yang, 2019]	2.85 ± 0.02	-	2.8	1.5	Gradient
GDAS[Dong and Yang, 2019]	2.93	-	3.4	0.2	Gradient
P-DARTS [Chen *et al.*, 2019]	2.50	16.55	3.4	0.3	Gradient
DARTS+PT [Wang *et al.*, 2021]	$2.48(2.61 \pm 0.08)$	$19.05^{\ddagger}$	3.0	0.8	Gradient
DARTS- [Chu *et al.*, 2020]	2.59 ± 0.08	-	3.5 ± 0.13	0.4	Gradient
R-DARTS(L2) [Zela *et al.*, 2019]	2.95 ± 0.21	-	-	1.6	Gradient
FairDARTS [Chu *et al.*, 2019]	2.54	-	2.8	0.4	Gradient
OLES	**2.29(2.38±0.11)**	**16.29(16.35 ± 0.05)**	3.4	0.4	Gradient

Table 1: Comparison with state-of-the-art image classifiers on CIFAR-10, and transfer to CIFAR-100. $^{\diamond}$: reported by [Chen *et al.*, 2019]. *: produced by rerunning the published code. ‡: transfer architectures discovered on CIFAR-10 to CIFAR-100.

Architecture	Test Err (%)	Params (M)	Search Cost (GPU-days)	Search Method
ResNet + CutOut[He *et al.*, 2015]	$22.10^{\diamond}$	1.7	-	Manual
PNAS [Liu *et al.*, 2017]	$19.53^{\diamond}$	3.2	150	MDL
ENAS + CutOut [Pham *et al.*, 2018]	$19.43^{\diamond}$	4.6	0.45	RL
DARTS [Liu *et al.*, 2018]	$20.58 \pm 0.44^{\dagger}$	3.4	0.4	Gradient
GDAS [Dong and Yang, 2019]	18.38	3.4	0.2	Gradient
P-DARTS [Chen *et al.*, 2019]	17.46^{*}	3.6	0.3	Gradient
DARTS- [Chu *et al.*, 2020]	17.51 ± 0.25	3.3	0.4	Gradient
R-DARTS(L2)[Zela *et al.*, 2019]	18.24^{*}	-	1.6	Gradient
OLES	**17.10**	3.4	0.4	Gradient

Table 2: Comparison of architectures on CIFAR-100. $^{\diamond}$: reported by [Dong and Yang, 2019]. †: reported by [Zela *et al.*, 2019]. *: produced by rerunning the published code.

are shown in Table 1. In this experiment, the threshold ξ is set to 0.4, and the patience K is set to 20. We repeat the search on CIFAR-10 three times with different random seeds while evaluating the resulting architectures on CIFAR-10 and CIFAR-100. And all the experiments are done on a Tesla V100 GPU.

As shown in Table 1, our method consistently outperforms other recent SOTA methods. OLES obtains a top-1 test error of 2.29% on CIFAR-10 with approximately the same search cost as DARTS of 0.4 GPU-days, which is a new SOTA. Moreover, the robustness of our method (OLES) is also guaranteed since the average result of three independent runs is outstanding as well. When the architectures searched on CIFAR-10 are transferred to CIFAR-100, our method still achieves the best test error of 16.29%. It indicates that the architectures searched by OLES are transferable and expressive, and the architecture parameters do not overfit the CIFAR-10 dataset.

CIFAR-100

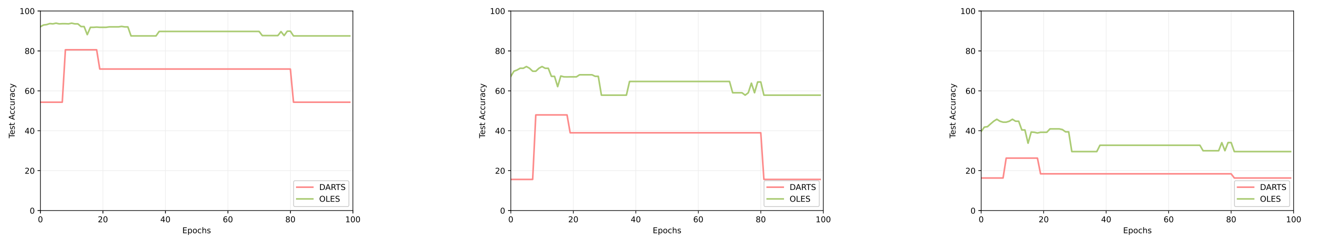
We also directly conduct the architecture search on CIFAR-100, and the results are presented in Table 2. The threshold ξ and patience K for CIFAR-100 are set to 0.3 and 20, respectively. Our method still exhibits significant advantages in this experiment, which improves the test error of DARTS from 20.58% to a new SOTA 17.10% without extra search cost. It demonstrates that our method can adapt to datasets of different sizes and consistently improve the performance of DARTS. Our OLES can make DARTS outperform other sophisticated differentiable architecture search methods without modifying much code or introducing extra computational overheads.

Transfer to ImageNet

To verify the transferability of architectures found by OLES, we transfer the best architecture derived from CIFAR-10 to ImageNet. ImageNet is the most commonly used dataset in the image classification, which contains $1,000$ object categories, and 1.28M training and 50K validation images. Following the setting of DARTS, we train the searched architecture for 250 epochs from scratch, and the results are shown in Table 3. Our method obtains 24.7% top-1 test error on the ImageNet validation dataset. Compared to other methods, architectures derived from OLES still have competitive

Architecture	Test Err (%)		Params (M)	Search Cost (GPU-days)	Search Method
	Top-1	Top-5			
Inception-v1 [Szegedy *et al.*, 2015]	30.2	10.1	6.6	-	Manual
MobileNet-V2 [Sandler *et al.*, 2018]	28.0	-	3.4	-	Manual
ShuffleNet [Zhang *et al.*, 2017]	26.3	-	∼5	-	Manual
NASNet-A [Zoph *et al.*, 2017]	26.0	8.4	5.3	1800	RL
AmoebaNet-A [Real *et al.*, 2018]	25.5	8.0	5.1	3150	Evolution
AmoebaNet-B [Real *et al.*, 2018]	26.0	8.5	5.3	3150	Evolution
PNAS [Liu *et al.*, 2017]	25.8	8.1	5.1	225	SMBO
MnasNet-92 [Tan *et al.*, 2018]	$25.2^{\diamond}$	$8.0^{\diamond}$	4.4	1667	RL
DARTS(2nd) + CutOut[Liu *et al.*, 2018]	26.9	8.7	4.7	0.4	Gradient
SNAS [Xie *et al.*, 2018]	27.3	9.2	4.3	1.5	Gradient
GDAS [Dong and Yang, 2019]	25.0	8.5	5.3	0.2	Gradient
P-DARTS [Chen *et al.*, 2019]	24.4	7.4	5.1	0.3	Gradient
PC-DARTS [Xu *et al.*, 2019]	25.1	7.8	5.3	3.8	Gradient
DARTS+PT [Wang *et al.*, 2021]	25.5	8.0	4.6	0.8	Gradient
FairDARTS-B [Chu *et al.*, 2019]	24.9	7.5	4.8	0.4	Gradient
NoisyDARTS-A [Chu and Zhang, 2020]	22.1	6.0	5.5	12	Gradient
DARTS- [Chu *et al.*, 2020]	22.8	6.1	5.5	4.5	Gradient
OLES	24.7	7.6	4.7	0.4	Gradient

Table 3: Comparison of architectures on ImageNet. $\diamond$: reported by [Chu *et al.*, 2019]. Other results are from their original papers. DARTS- and NoisyDARTS-A are directly searched on ImageNet and use SE modules and Swish to improve performance. Other methods in the bottom block are searched on CIFAR-10 and then transferred to ImageNet.



(a) Test accuracy on CIFAR-10. (b) Test accuracy on CIFAR-100. (c) Test accuracy on ImageNet16-120.

Figure 3: Trajectory of test accuracy on space NAS-Bench-201 and three datasets.

performance on ImageNet. Note that we just follow the original DARTS setting to transfer the architectures to ImageNet without introducing any extra training tricks such as SE modules and Swish. Therefore, the performance of OLES on ImageNet can be further enhanced by adding these training tricks. It further demonstrates the transferability of architectures derived by OLES. The architectures searched on CIFAR-10 not only achieve the state-of-the-art on CIFAR-10 and CIFAR-100 but also perform well on ImageNet.

4.3 Performance on NAS-Bench-201

NAS-Bench-201 [Dong and Yang, 2020] guarantees a unified benchmark for analyzing various up-to-date NAS algorithms. It contains 4 internal nodes with 5 operations (i.e., Zero, Skip Connection, 1×1 Conv, 3×3 Conv, 3×3 AvgPol),. And NAS-Bench-201 offers a similar cell-based search space that is comprised of a total of 15625 unique architectures. The architectures are trained under three datasets (i.e., CIFAR-10, CIFAR-100 and ImageNet16-120). We perform DARTS and OLES on the NAS-Bench-201 and record the test accuracy of the searched architecture in each training epoch. The results are shown in Figure 3. DARTS performs extremely worse on NAS-Bench-201. We also observe that the skip connection quickly dominates the architectures searched by DARTS in around 5 epochs, causing the performance collapse. We suppose that it is due to the fact that the search space of NAS-Bench-201 is extremely compact, contains few parametric operations, and the number of operation parameters varies widely, which makes it difficult for DARTS to adapt to NAS-Bench-201.

OLES significantly outperforms DARTS on all datasets and search epochs. It can demonstrate the generality of OLES, which can adapt to different search spaces and datasets. However, the performance of OLES is still far from optimal. OLES is implemented on the basis of DARTS, and the experimental settings of DARTS also limit the performance of OLES to some extent. Furthermore, the compact search space makes tuning OLES more difficult, as it will be more sensitive to the threshold ξ. In the future, we will investigate the underlying reason for why DARTS collapses on

NAS-Bench-201, and optimize OLES to achieve SOTA results on NAS-Bench-201.

4.4 Sensitivity of Threshold ξ and Patience K

Since OLES mainly introduces two hyperparameters, the overfitting threshold ξ and the patience K, for determining whether the operation tends is overfitting, we conduct experiments to analysis their impact. For CIFAR-10, we select the threshold in $\{0.4, 0.5, 0.6\}$ and count the largest number of steps (i.e. K_{up}) in which the difference between operation directions consistently exceeds corresponding threshold. The results are shown in Table 4. When the threshold ξ is set to 0.4, the difference between operation directions will consistently exceed 0.4, up to 63 consecutive steps. However, when the threshold ξ is set to 0.6, the number is dropped to 1. It demonstrates the effectiveness of OLES that the difference between operation gradients does indeed relate to overfitting in training. And OLES is very sensitive to the threshold ξ which needs to be tuned carefully.

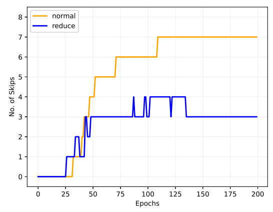
To find the best hyperparameters, we firstly determine the threshold ξ and then train DARTS several epochs to record the upper steps K_{upper}. Then we set the interval of patience based on K_{upper}. Finally, we tune the patience K to find the best hyperparameter settings. We also report the best patience K in Table 4. OLES can achieve the best result on CIFAR-10 with threshold $\xi = 0.4$ and patience $K = 20$. In the implementation, it is easy to determine a set of hyperparameters based on experience, so that OLES can achieve comparable results. We will explore finer-grained hyperparameter spaces and ways to automatically determine the hyperparameters in the future.

Threshold ξ	Upper Steps K_{upper}	Patience K	Test Error(%)
0.4	63	20	2.29
0.5	5	4	2.73
0.6	1	1	2.88

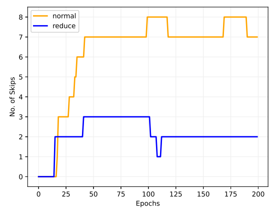
Table 4: Test errors evaluated on CIFAR-10 with different overfitting threshold ξ and patience K. The upper steps K_{upper} means the largest number of steps in which the difference between operation directions exceeds the corresponding threshold.

4.5 Training With Longer Epochs

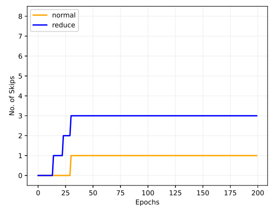
As mentioned in [Bi *et al.*, 2019] and [Chu *et al.*, 2020], training with longer epochs enables a better convergence of the supernet and architecture parameters. However, as shown in Figure 4a and Figure 4b, training standard DARTS with longer epochs on CIFAR-10 and CIFAR-100 shows serious aggregation of skip connections. The skip connection rapidly dominates after 50 epochs on CIFAR-10 and takes longer epochs on CIFAR-100. This phenomenon can also verify our conjecture of overfitting. We also find that other methods (e.g. Fair DARTS, and DARTS-PT) will fail with longer training epoch as well. More results and corresponding genotypes can be found in the appendix. As shown in Figure 4, for both datasets, OLES survives after 200 epochs as expected without the domination of skip connection. The number of skip connections in OLES keeps stable after a specific number of epochs since most parametric operations will be frozen and not affect the training of architecture parameters. It demonstrates that OLES is robust and can actually solve the domination of skip connection problem.



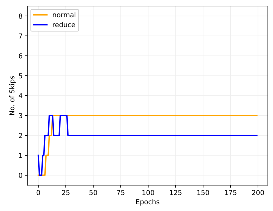
(a) The number of skip connections when searching with DARTS on CIFAR-10 dataset.



(b) The number of skip connections when searching with DARTS on CIFAR-100 dataset.



(c) The number of skip connections when searching with OLES on CIFAR-10 dataset.



(d) The number of skip connections when searching with OLES on CIFAR-100 dataset.

Figure 4: Trajectories of the numbers of skip connections on CIFAR-10 and CIFAR-100.

5 Conclusion and Future Work

This paper attempts to find the fundamental reason for the domination of skip connection in DARTS from the new perspective of overfitting of operations in the supernet. We verified our conjecture through a motivational experiment and proposed the operation-level early stopping (OLES) method based on this conjecture to solve the domination of skip connection. OLES monitors the gradient direction of each operation and determine whether it is overfitting based on the difference between the gradient directions on the training data and the validation data. Overfitting operations will stop training during the supernet training. The proposed method can solve the domination of skip connection with negligible computational overheads and without imposing any limitations. Moreover, OLES achieves SOTA results of 2.29% test error on CIFAR10 and 17.10% test error on CIFAR-100, meanwhile reporting superior performance when transferred to ImageNet.

In summary, we proposed a new perspective for understanding the robustness of differentiable NAS. We hope our work can inspire future works exploring this field. In the future, we will try to refine our conjecture from a theoretical perspective, further optimize OLES, and validate it on a wider range of search spaces and datasets.

References

[Bi *et al.*, 2019] Kaifeng Bi, Changping Hu, Lingxi Xie, Xin Chen, Longhui Wei, and Qi Tian. Stabilizing darts with amended gradient estimation on architectural parameters, 2019.

[Cai *et al.*, 2018] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018.

[Chen *et al.*, 2019] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive darts: Bridging the optimization gap for nas in the wild, 2019.

[Chu and Zhang, 2020] Xiangxiang Chu and Bo Zhang. Noisy differentiable architecture search, 2020.

[Chu *et al.*, 2019] Xiangxiang Chu, Tianbao Zhou, Bo Zhang, and Jixiang Li. Fair darts: Eliminating unfair advantages in differentiable architecture search, 2019.

[Chu *et al.*, 2020] Xiangxiang Chu, Xiaoxing Wang, Bo Zhang, Shun Lu, Xiaolin Wei, and Junchi Yan. Darts-: Robustly stepping out of performance collapse without indicators, 2020.

[Dong and Yang, 2019] Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours, 2019.

[Dong and Yang, 2020] Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations*, 2020.

[Guo *et al.*, 2020] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *European Conference on Computer Vision*, pages 544–560. Springer, 2020.

[He *et al.*, 2015] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[Huang *et al.*, 2016] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks, 2016.

[Li and Talwalkar, 2020] Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. In *Uncertainty in artificial intelligence*, pages 367–377. PMLR, 2020.

[Liang *et al.*, 2019] Hanwen Liang, Shifeng Zhang, Jiacheng Sun, Xingqiu He, Weiran Huang, Kechen Zhuang, and Zhenguo Li. Darts+: Improved differentiable architecture search with early stopping, 2019.

[Liu *et al.*, 2017] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search, 2017.

[Liu *et al.*, 2018] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search, 2018.

[Liu *et al.*, 2019] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 82–92, 2019.

[Nayman *et al.*, 2019] Niv Nayman, Asaf Noy, Tal Ridnik, Itamar Friedman, Rong Jin, and Lihi Zelnik. Xnas: Neural architecture search with expert advice. *Advances in neural information processing systems*, 32, 2019.

[Pham *et al.*, 2018] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing, 2018.

[Real *et al.*, 2018] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search, 2018.

[Sandler *et al.*, 2018] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. 2018.

[So *et al.*, 2019] David So, Quoc Le, and Chen Liang. The evolved transformer. In *International Conference on Machine Learning*, pages 5877–5886. PMLR, 2019.

[Szegedy *et al.*, 2015] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015.

[Tan *et al.*, 2018] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. Mnasnet: Platform-aware neural architecture search for mobile. 2018.

[Termritthikun *et al.*, 2021] Chakkrit Termritthikun, Yeshi Jamtsho, Jirarat Ieamsaard, Paisarn Muneesawang, and Ivan Lee. Eeea-net: An early exit evolutionary neural architecture search. *Engineering Applications of Artificial Intelligence*, 104:104397, 2021.

[Wang *et al.*, 2021] Ruochen Wang, Minhao Cheng, Xiangning Chen, Xiaocheng Tang, and Cho-Jui Hsieh. Rethinking architecture selection in differentiable nas, 2021.

[Xie *et al.*, 2018] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926*, 2018.

[Xu *et al.*, 2019] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. Pc-darts: Partial channel connections for memory-efficient architecture search, 2019.

[Yu *et al.*, 2019] Kaicheng Yu, Christian Sciuto, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. Evaluating the search phase of neural architecture search. *arXiv preprint arXiv:1902.08142*, 2019.

[Zela *et al.*, 2019] Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. Understanding and robustifying differentiable architecture search, 2019.

[Zhang *et al.*, 2017] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices, 2017.

[Zoph and Le, 2016] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

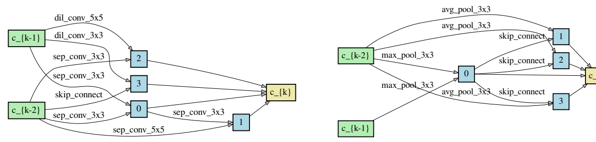
[Zoph *et al.*, 2017] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition, 2017.

A Training with Longer Epochs

We train OLES and several other methods with 200 epochs to verify whether they are robust enough in this case. According to our conjecture, as the number of training epochs of DARTS increases, the overfitting of operations in the supernet will be more serious, and thus the domination of skip connection problem will be more serious. In this section, we present some additional results of the searched architectures at different epochs.

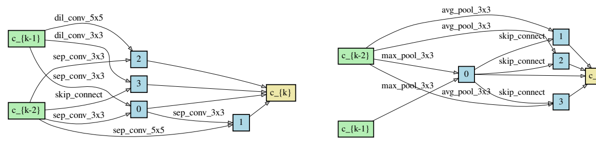
A.1 Results on OLES

As described in Section 4.5, our method (OLES) can survive with longer epochs. The number of skip connections in OLES keeps stable when the number of training epochs reaches 150 and 200 epochs. The corresponding architecture genotypes on CIFAR-10 and CIFAR-100 are visualized in Figure 5, 6, 7 and 8.



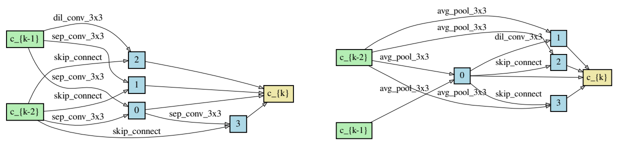
(a) Normal Cell (b) Reduction Cell

Figure 5: Normal and Reduction cells discovered by OLES on CIFAR-10 for 150 epochs in the standard DARTS' search space.



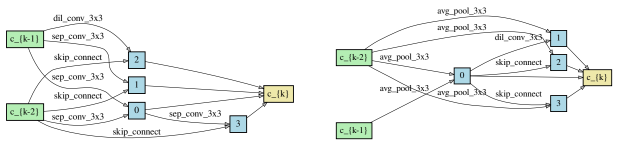
(a) Normal Cell (b) Reduction Cell

Figure 6: Normal and Reduction cells discovered by OLES on CIFAR-10 for 200 epochs in the standard DARTS' search space.



(a) Normal Cell (b) Reduction Cell

Figure 7: Normal and Reduction cells discovered by OLES on CIFAR-100 for 150 epochs in the standard DARTS' search space.



(a) Normal Cell (b) Reduction Cell

Figure 8: Normal and Reduction cells discovered by OLES on CIFAR-100 for 200 epochs in the standard DARTS' search space.

A.2 Results on FairDARTS

FairDARTS uses the *Sigmoid* instead of the exclusive *Softmax* and selects operations through a threshold value. We found that no operation can be obtained in the normal cell with the threshold value when FairDARTS is trained for 150 or 200 epochs. Following FairDARTS' settings, we train it for 50, 150, and 200 epochs. Table 5 shows the architecture genotypes, which can be visualized in Figure 9 and Figure 10.

A.3 Results on DARTS+PT

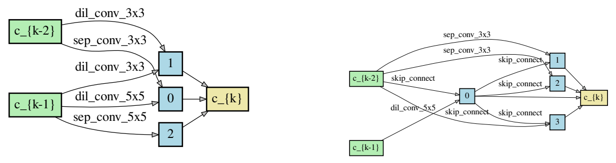
DARTS+PT directly evaluates operation strength in terms of its contribution to the supernet's performance instead of the magnitude of the architecture parameter. Following DARTS+PT's settings, we train DARTS with the permutation-based architecture selection method for 50, 150, and 200 epochs, and the corresponding architecture genotypes are shown in Table 6. When extending to over 150 epochs, the resulting architectures are also full of skip connections. Moreover, DARTS+PT can tune the supernet when discretizing operations with additional epochs to make the network converge again. DARTS+PT both tunes the supernets trained for 150 and 200 epochs for additional 80 epochs. The architecture genotypes with tuned supernets are shown in Table 7, Figure 11 and 12. By the way, the test error of the final resulting architecture for 200 epochs will drop to 6.28%.

Epochs	Architecture genotype
50	Genotype(normal=[(’sep_conv_3x3’, 2, 0), (’dil_conv_5x5’, 2, 1), (’dil_conv_3x3’, 3, 0), (’dil_conv_3x3’, 3, 1), (’sep_conv_5x5’, 4, 1)], normal_concat = range(2, 6), reduce= [(’skip_connect’, 2, 0), (’dil_conv_5x5’, 2, 1), (’skip_connect’, 3, 2), (’sep_conv_3x3’, 3, 0), (’skip_connect’, 4, 2), (’sep_conv_3x3’, 4, 0), (’skip_connect’, 5, 2), (’skip_connect’, 5, 0)], reduce_concat=range(2, 6))
150	Genotype(normal=[ ], normal_concat=range(2, 6), reduce=[(’max_pool_3x3’, 2, 0), (’max_pool_3x3’, 2, 1), (’skip_connect’, 3, 2), (’max_pool_3x3’, 3, 0), (’skip_connect’, 4, 2), (’max_pool_3x3’, 4, 0), (’skip_connect’, 5, 3), (’skip_connect’, 5, 2)], reduce_concat= range(2, 6))
200	Genotype(normal=[ ], normal_concat=range(2, 6), reduce=[(’avg_pool_3x3’, 2, 0), (’ max__pool_3x3’, 2, 1), (’skip_connect’, 3, 2), (’max_pool_3x3’, 3, 0), (’skip_connect’, 4, 3), (’skip_connect’, 4, 2), (’sep_conv_3x3’, 5, 0), (’dil_conv_3x3’, 5, 1)], reduce_concat= range(2, 6))

Table 5: Architecture genotypes found by FairDARTS on CIFAR-10 for 50, 150 and 200 epochs, respectively.

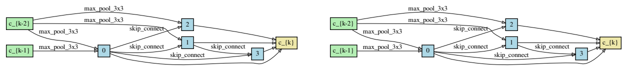
Epochs	Architecture genotype
50	Genotype(normal=[(’sep_conv_3x3’, 0), (’dil_conv_3x3’, 1), (’sep_conv_3x3’, 0), (’dil_conv_3x3’, 2), (’dil_conv_3x3’, 3), (’skip_connect’, 0), (’dil_conv_5x5’, 4), (’dil_conv_5x5’,3)], normal_concat=range(2, 6), reduce=[(’max_pool_3x3’, 0), (’max_pool_3x3’, 1), (’max_pool_3x3’, 0), (’skip_connect’, 2), (’skip_connect’, 2), (’skip_connect’, 3), (’skip_connect’, 2), (’avg_pool_3x3’, 0)], reduce_concat=range(2, 6))
150	Genotype(normal=[(’dil_conv_3x3’, 1), (’sep_conv_3x3’, 0), (’skip_connect’, 2), (’skip_connect’, 0), (’skip_connect’, 1), (’skip_connect’, 2), (’skip_connect’, 1), (’skip_connect’, 2)], normal_concat=range(2, 6), reduce=[(’max_pool_3x3’, 1), (’max_pool_3x3’, 0), (’skip_connect’, 2), (’max_pool_3x3’, 0), (’skip_connect’, 2), (’skip_connect’, 3), (’skip_connect’, 3), (’skip_connect’, 2)], reduce_concat=range(2, 6))
200	Genotype(normal=[(’dil_conv_3x3’, 1), (’sep_conv_3x3’, 0), (’skip_connect’, 2), (’skip_connect’, 1), (’skip_connect’, 1), (’skip_connect’, 2), (’skip_connect’, 1), (’skip_connect’, 0)], normal_concat=range(2, 6), reduce=[(’max_pool_3x3’, 1), (’avg_pool_3x3’, 0), (’skip_connect’, 2), (’avg_pool_3x3’, 0), (’skip_connect’, 2), (’skip_connect’, 3), (’skip_connect’, 3), (’skip_connect’, 2)], reduce_concat=range(2, 6))

Table 6: Architecture genotypes found by DARTS+PT on CIFAR-10 for 50, 150 and 200 epochs, respectively. (without tuning)



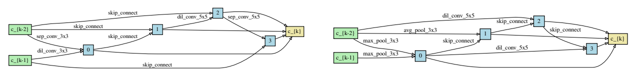
(a) Normal Cell (b) Reduction Cell

Figure 9: Normal and Reduction cells discovered by FairDARTS on CIFAR-10 for 50 epochs in the standard DARTS’ search space.



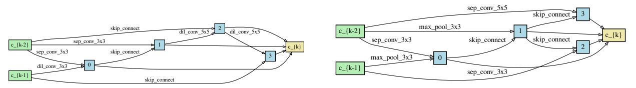
(a) Epoch=150 (b) Epoch=200

Figure 10: Reduction cells discovered by FairDARTS on CIFAR-10 for 150 and 200 epochs in the standard DARTS’ search space. Note that normal cells will be empty when FairDARTS is trained for 150 and 200 epochs.



(a) Normal Cell (b) Reduction Cell

Figure 11: Normal and Reduction cells discovered by DARTS+PT on CIFAR-10 for 150 epochs in the standard DARTS’ search space. (with tuning)



(a) Normal Cell (b) Reduction Cell

Figure 12: Normal and Reduction cells discovered by DARTS+PT on CIFAR-10 for 200 epochs in the standard DARTS’ search space. (with tuning)

Epochs	Architecture genotype
150	Genotype(normal=[('sep_conv_3x3', 0), ('dil_conv_3x3', 1), ('skip_connect', 0), ('skip_connect', 2), ('skip_connect', 0), ('dil_conv_5x5', 3), ('skip_connect', 1), ('sep_conv_5x5', 4)], normal_concat=range(2, 6), reduce=[('max_pool_3x3', 0), ('max_pool_3x3', 1), ('avg_pool_3x3', 0), ('skip_connect', 2), ('dil_conv_5x5', 0), ('skip_connect', 3), ('dil_conv_5x5', 2), ('skip_connect', 4)], reduce_concat=range(2, 6))
200	Genotype(normal=[('sep_conv_3x3', 0), ('dil_conv_3x3', 1), ('sep_conv_3x3', 0), ('skip_connect', 2), ('skip_connect', 0), ('dil_conv_5x5', 3), ('skip_connect', 1), ('dil_conv_5x5', 4)], normal_concat=range(2, 6), reduce=[('sep_conv_3x3', 0), ('max_pool_3x3', 1), ('max_pool_3x3', 0), ('skip_connect', 2), ('sep_conv_3x3', 1), ('skip_connect', 3), ('sep_conv_5x5', 0), ('skip_connect', 3)], reduce_concat=range(2, 6))

Table 7: Architecture genotypes found by DARTS+PT on CIFAR-10 for 50, 150 and 200 epochs, respectively. (with tuning)