DARTS+: Improved Differentiable Architecture Search with Early Stopping

Hanwen Liang[1*] **Shifeng Zhang**[2*] **Jiacheng Sun**[1†] **Xingqiu He**[1]
Weiran Huang[1] **Kechen Zhuang**[1] **Zhenguo Li**[1]

[1]Huawei Noah's Ark Lab [2]TNList, Tsinghua University

Abstract

Recently, there has been a growing interest in automating the process of neural architecture design, and the Differentiable Architecture Search (DARTS) method makes the process available within a few GPU days. However, the performance of DARTS is often observed to collapse when the number of search epochs becomes large. Meanwhile, lots of "*skip-connect*s" are found in the selected architectures. In this paper, we claim that the cause of the collapse is that there exists overfitting in the optimization of DARTS. Therefore, we propose a simple and effective algorithm, named "DARTS+", to avoid the collapse and improve the original DARTS, by "early stopping" the search procedure when meeting a certain criterion. We also conduct comprehensive experiments on benchmark datasets and different search spaces and show the effectiveness of our DARTS+ algorithm, and DARTS+ achieves 2.32% test error on CIFAR10, 14.87% on CIFAR100, and 23.7% on ImageNet. We further remark that the idea of "early stopping" is implicitly included in some existing DARTS variants by manually setting a small number of search epochs, while we give an *explicit* criterion for "early stopping".

1 Introduction

Neural Architecture Search (NAS) plays an important role in Automated Machine Learning (AutoML), which has attracted a lot of attention recently [3, 8, 20–22, 26, 35, 42]. Differentiable Architecture Search (DARTS) [21] receives broad attention as it can perform searching very fast while achieves the desired performance. In particular, it encodes the architecture search space with continuous parameters to form a one-shot model and performs searching by training the one-shot model with gradient-based bi-level optimization.

Despite the efficiency of DARTS, a critical issue of DARTS has been found [3, 4, 37, 41]. Namely, after certain search epochs, the number of *skip-connect*s increases dramatically in the selected architecture, which results in poor performance. We call the phenomenon of performance drop after a certain number of epochs the "collapse" of DARTS.

To tackle such an issue, some works like P-DARTS [3] design search space regularization to alleviate the dominance of *skip-connect*s during the search. However, these approaches involve more hyper-parameters, which need to be carefully tuned by human experts. Moreover, Single-Path NAS [30], StacNAS [17], and SNAS [34] use the one-level optimization instead of the bi-level optimization in DARTS, where the architecture parameters and model weights are updated simultaneously. However, the search spaces of these algorithms need to be carefully designed [17, 21, 34]. In summary, the mechanism of the collapse of DARTS remains open.

In this paper, we first show that the collapse of DARTS is due to the *overfitting* in the search phase, which results in a large gap between training and validation error. In particular, we explain why *overfitting* results in a large number of *skip-connect*s in the selected architectures in DARTS, which hurts the performance of the selected architecture. To avoid the collapse of DARTS, we add a simple and effective "early stopping" paradigm, called "DARTS+", where the search procedure stops by a certain criterion, illustrated in Fig. 1.(a). We point out that the searching is saturated when it is "early stopped". We remark that some progress of DARTS, including P-DARTS [3], Auto-DeepLab [19], and PC-DARTS [36], also adopt the early stopping idea implicitly where fewer search epochs are manually set in their methods.

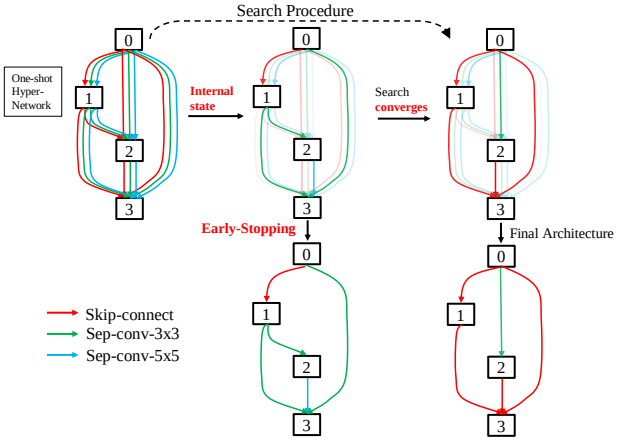
Moreover, we conduct sufficient experiments to demonstrate the effectiveness of the proposed DARTS+ algorithm. Specifically, DARTS+ succeeds in searching on various spaces including DARTS, MobileNetV2, ResNet. In the DARTS search space, DARTS+ achieves 2.32% test error on CIFAR10 and 14.87% test error on CIFAR100, while the search time is less than 0.4 GPU days. When transferring to ImageNet, DARTS+ achieves 23.7% top-1 error and impressive 22.5% top-1 error if SE-Module [12] is introduced. DARTS+ is also able to search on ImageNet directly and gets 23.9% top-1 error.

In summary, our main contributions are as follows:

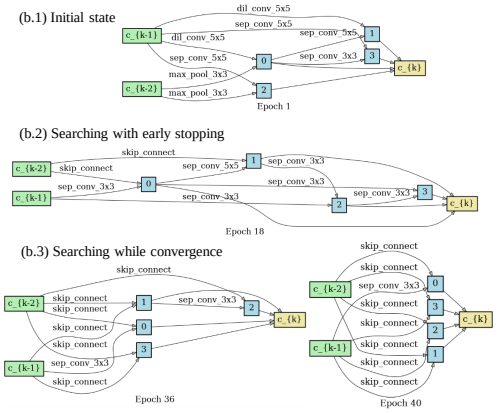
- We study the collapse issue of DARTS, and point out the underlying reason is the overfitting of the model weights in the DARTS training.

[*]Equal contribution. This work was done when the first two authors were interns at Huawei Noah's Ark Lab.

[†]Corresponding email: sunjiacheng1@huawei.com.



(a) Illustration on early stopping in DARTS



(b) Selected architectures at different search epochs on CIFAR100

Figure 1: (a) Illustration of the early stopping paradigm. (b) Selected architectures at different search epochs on CIFAR100. With different stopping mechanisms, we may obtain better architectures or end up with collapsed architectures with lots of skip-connects.

- We introduce an efficient "early stopping" paradigm for DARTS to avoid the collapse and propose effective and adaptive criteria for early stopping.
- We conduct extensive experiments on benchmark datasets and various search spaces to demonstrate the effectiveness of the proposed algorithm, which achieves state-of-the-art results on all of them.

2 Collapse of DARTS

There is an undesired behavior of DARTS [21] that too many *skip-connect*s tend to appear in the selected architecture when the number of search epochs is large, making the performance poor. The phenomenon of performance drop is called the "collapse" of DARTS in our paper. In this section, we first give a quick review of the original DARTS, and then point out the collapse issue of DARTS and discuss its underlying causes.

2.1 DARTS

The goal of DARTS is to search for a cell, which can be stacked to form a convolutional network or a recurrent network. Each cell is a directed acyclic graph (DAG) of N nodes $\{x_i\}_{i=0}^{N-1}$, where each node represents a network layer. We denote the operation space as $\mathcal{O}$, and each element is a candidate operation, e.g., *zero, skip-connect, convolution, max-pool*, etc. Each edge (i,j) of DAG represents the information flow from node x_i to x_j, which consists of the candidate operations weighted by the architecture parameter $\alpha^{(i,j)}$. In particular, each edge (i,j) can be formulated by a function $\bar{o}^{(i,j)}$ where $\bar{o}^{(i,j)}(x_i) = \sum_{o\in\mathcal{O}} p_o^{(i,j)} \cdot o(x_i)$, and the weight of each operation $o \in \mathcal{O}$ is a softmax of the architecture parameter $\alpha^{(i,j)}$, that is $p_o^{(i,j)} = \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o'\in\mathcal{O}} \exp(\alpha_{o'}^{(i,j)})}$. An intermediate node is $x_j = \sum_{i<j} \bar{o}^{(i,j)}(x_i)$, and the output node x_{N-1} is depth-wise concatenation of all the intermediate nodes excluding input nodes. The above hyper-network is called the one-shot model, and we denote w as the weights of the hyper-network.

For the search procedure, we denote $\mathcal{L}_{train}$ and $\mathcal{L}_{val}$ as the training and validation loss respectively. Then the architecture parameters are learned with the following bi-level optimization problem:

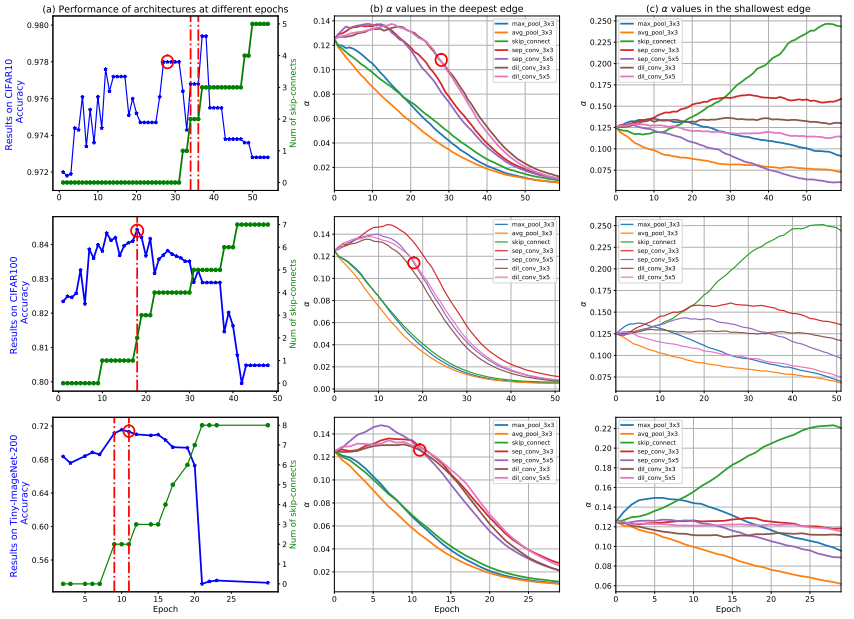
$$\begin{aligned} \min_{\alpha} \quad & \mathcal{L}_{val}(w^*(\alpha), \alpha), \\ \text{s.t.} \quad & w^*(\alpha) = \arg\min_{w} \mathcal{L}_{train}(w, \alpha). \end{aligned}$$

After obtaining architecture parameters α, the final discrete architecture is derived by: 1) setting $o^{(i,j)} = \arg\max_{o\in\mathcal{O}, o\neq zero} p_o^{(i,j)}$, and 2) for each intermediate node, choosing two incoming edges with the two largest values of $\max_{o\in\mathcal{O}, o\neq zero} p_o^{(i,j)}$. More technical details can be found in the original DARTS paper [21].

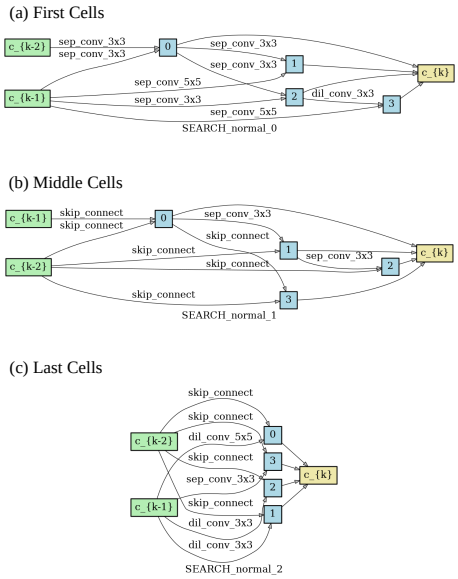
2.2 Collapse Issue

It has been observed in DARTS that lots of *skip-connects* are involved in the selected architecture, which makes the architecture shallow and the performance poor. As an example, let us consider searching on CIFAR100. The α value of *skip-connect*s (green line in Fig. 2(i)(c)) becomes large when the number of search epochs is large, thus the number of *skip-connect*s increases in the selected architecture as shown in the green line in Fig. 2(i)(a). Such a shallow network has less learnable parameters than deep ones, and thus it has weaker expressive power. As a result, architectures with lots of *skip-connect*s have poor performance, i.e., *collapsed*, indicated as the blue line in Fig. 2(i)(a)[1], see more experiments in Appendix C. To be more intuitive, we draw the selected architectures from different search epochs on CIFAR100 in

[1] All the architecture are fully trained and evaluated in the same settings.



(i) Detailed illustration on the collapse of DARTS. (a) The performance of architectures at different epochs on CIFAR10, CIFAR100, and Tiny-ImageNet-200, respectively (in blue line), and the number of *skip-connect*s in the normal cell (in green line). (b) The change of α in the deepest edge (connecting the last two nodes) of the one-shot model. We omit the α of *none* operation as it increases while α of other operations drops. (c) The change of architecture parameters α in the shallowest edge. The dashed line denotes the early-stopping paradigm introduced in Sec. 3, and the circle denotes the point that the α ranking of learnable parameters becomes stable.



(ii) The selected architecture of normal cells at different layers when searching distinct cell architectures in different stages (stages are split with reduction cells). The searched dataset is CIFAR100. The first cells contain mostly convolutions, while the last cells are shallow with numerous *skip-connect*s.

Figure 2: The collapse issue of DARTS.

Fig. 1(b). When the number of search epochs increases, the number of *skip-connect*s in the selected architecture also increases. Such a phenomenon can also be observed on other datasets, such as CIFAR10 and Tiny-ImageNet-200.

To avoid the collapse, one might propose to adjust searching hyper-parameters, such as 1) adjusting learning rates, 2) changing the portion of training and validation data, and 3) adding regularization on *skip-connect*s like *dropout*. Unfortunately, such methods only alleviate the collapse at certain searching epochs but the collapse would finally appear, implying that the choice of hyperparameters is not the essential cause of the collapse.

2.3 Overfitting and Analysis

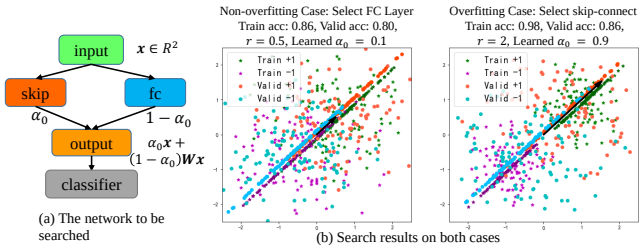
To figure out the collapse of DARTS, we observe that the model weights in the one-shot model suffer from *"overfitting"* during the search procedure. In the bi-level optimization of DARTS, the model weights w are updated with the training data while the architecture parameters α are updated with the validation data. As the model weights, w in the one-shot model is over-parameterized, w tends to fit the training data well, while the validation data is underfitted as the number of α is limited. To be specific, in CIFAR10/100 dataset, the training accuracy can reach 99% while the validation accuracy is just 88% in CIFAR10 and 60% in CIFAR100. This implies *"overfitting"* as the gap between the training and validation error is large.[2]

We show that the *overfitting* of the model weights is the main cause of the collapse of DARTS. In particular, at the initial state, the model weights underfit the training data and the gap between the training and validation error is small. Therefore, the architecture parameters α and model weights w get better together. After certain search epochs, the model weights overfit the training data. However, on validation data, they fit not as well as training data and the first cells of the model could obtain relatively better low-level feature representations than those in the last cells.

If we allow different cells to have distinct architectures in the one-shot model, the last cells are more likely to select more *skip-connect*s to obtain the good feature representation directly from the first cells. Fig. 2(ii) shows the learned normal cell architectures at different layers if we search different architectures at different stages[3]. It can be seen that the

[2]The definition of "overfitting" is slightly different from the general definition such that lower training error results in a higher validation error. However, in both cases, the gap between the training and validation error is large.

[3]Stages are split with reduction cells, and each stage consists of several stacked cells.



(a) The network to be searched

(b) Search results on both cases

Figure 3: Illustration of overfitting on the synthetic data. (a) The network to be searched, in which the detailed network definition is in Lemma 1. (b) Search results in both cases. The training/validation input features are a mixture of Gaussians, the features after the fully-connect(fc) layer are along the straight line, and the black arrow denotes $\mathbf{w}_r = r\mathbf{e}$.

algorithm tends to select deep architectures with learnable operations in the first cells (Fig. 2(ii)(a)), while architectures with many *skip-connect*s are preferred in the last cells (Fig. 2(ii)(c)). It accords with the previous analysis such that the last layers will select *skip-connect*s. If different cells are forced to have the same architecture, as DARTS does, continuous searching and fitting will push *skip-connect*s to be broadcast from the last cells to the first cells, making the number of *skip-connect*s in the selected architecture grow gradually. We could see the overfitting of model weights causes the degradation of the selected model architecture.

This *over-fitting* phenomenon can be further illustrated as a synthetic binary classification problem shown in Fig. 3. The one-shot model is a 2-layer network defined as $o(\mathbf{x}) = \mathbf{w}_r^\top(\alpha_0\mathbf{x} + (1-\alpha_0)\mathbf{W}\mathbf{x})$, where $\mathbf{W}, \mathbf{w}_r$ are model weights with $\|\mathbf{w}_r\| = r$, and α_0 is the architecture parameter (shown in Fig. 3(a)). The training data $\mathcal{T}$ and validation data $\mathcal{V}$ used for architecture search are 2-D feature representations and they are mixture of Gaussians such that $\mathcal{T} = \{(\mathbf{x}_i, y_i), y_i\mathbf{x}_i \sim N(\mu_t\mathbf{e}, \sigma_t^2\mathbf{I})\}$, $\mathcal{V} = \{(\mathbf{x}_i, y_i), y_i\mathbf{x}_i \sim N(\mu_v\mathbf{e}, \sigma_v^2\mathbf{I})\}$ where $\mathbf{e} = \frac{1}{\sqrt{2}}(1,1)^\top$. Both training and validation labels are balanced such that the number of data with label 1 is the same as that with label -1. If the one-shot model is searched with DARTS where the training and validation losses are $\mathcal{L}_{train} = \sum_{(\mathbf{x}_i,y_i)\in\mathcal{T}} l(o(\mathbf{x}_i), y_i), \mathcal{L}_{val} = \sum_{(\mathbf{x}_i,y_i)\in\mathcal{V}} l(o(\mathbf{x}_i), y_i), l(o,y) = \log(1+\exp(-yo))$, then under certain conditions, *skip-connect* will be selected, which is summarized in the following lemma.

Lemma 1 *Consider searching with a binary classification problem where the data and the one-shot model are defined above (shown in Fig. 3). Suppose (1) the feature representations are normalized such that* $\frac{1}{2}\mu_t^2 + \sigma_t^2 = \frac{1}{2}\mu_v^2 + \sigma_v^2 = 1$ [4]*, and the outputs of the fully-connected (fc) layer* $\{\mathbf{W}\mathbf{x}\}$ *are normalized as above; (2) the above losses are defined and are trained with bi-level optimization. Then we have*

***P1**: If* σ_t *is small, for any* α_0, $\mathbf{w}_r \to r\mathbf{e}, \mathbf{W} \to \mathbf{W}^*$, *where* $\mathbf{W}^*\mathbf{x} = t_\mathbf{x}\mathbf{e}$, *and* $t_\mathbf{x} = \frac{2}{\sqrt{2+\mu_t^2}}\mathbf{e}^\top\mathbf{x}$.

***P2** If* σ_t *is small and* $\sigma_v > \sigma_0(r)$ *where* $\sigma_0(r)$ *is a monotonic decreasing function, then* $\frac{\mathrm{d}\mathcal{L}_{val}}{\mathrm{d}\alpha_0} < 0$, *which implies that* α_0 *will become larger with gradient descent.*

The proof can be found in Appendix A. In this paper, the feature representations discussed in Lemma 1 correspond to the last layer feature representations in the one-shot model. To be specific, in the beginning of the search procedurem σ_v are large and r is small; When overfitting occurs, r goes larger to make the training data more separable while σ_v stays large. σ_t tend to be small during searching. According to Lemma 1, in the beginning of searching, σ_v is large while r is small and therefore $\sigma_v < \sigma_0(r)$, then learnable operations is preferred (left figure in Fig. 3(b)). When overfitting occurs, σ_v stays relatively large such that $\sigma_v > \sigma_0(r)$ as larger r makes $\sigma_0(r)$ small, then *skip-connect* tend to be selected (right figure in Fig. 3(b)).

3 The Early Stopping Methodology

Since the collapse issue of DARTS is caused by *"overfitting"* of the one-shot model in the bi-level optimization as pointed out in Sec. 2.2, we propose a simple and effective "early stopping" paradigm based on DARTS to avoid the collapse. In particular, the search procedure should be early stopped at an adaptive criterion, when DARTS starts to collapse. Such a paradigm leads to both better performance and fewer search costs than the original DARTS. We use DARTS+ to denote the DARTS algorithm with our early stopping criterion.

We want to emphasize that early stopping is essential and more attention should be paid. It has been found that important connections are determined in the early phase of training [1]. Significant and consequential changes occur during the earliest stage of training [7].

Besides the *"overfitting"* issue, another motivation of "early stopping" is that the ranking of architecture parameters αs of operations matters as only the operation with the maximum α is chosen in the selected architecture. During searching, the validation data has different preferences on learnable operations, which corresponds to the ranking of α values. If the rank of α is not stable, the architecture is too noisy to be chosen; while when it becomes stable, the learnable operations in the final selected architecture are not changed, and we could consider this point as the saturated search point. The red circles in Fig. 2(i)(a-b) denote the saturated search points on different datasets. It verifies that after this point the validation accuracies of selected architectures on all datasets (blue lines) tend to decrease, i.e., collapse. To conclude, the search procedure can be "early-stopped" at the saturated searching point to select desired architectures as well as avoiding overfitting, and we emphasize that this point does not mean the convergence of the one-shot model.

We first of all follow the cell-based architecture used by DARTS. The first criterion is stated as follows.

[4] Denote $\mathbf{x}[i]$ as the ith dimension of $\mathbf{x}$. As the training features are normalized, for the training data, it should be $\mathbb{E}_\mathcal{T}[\mathbf{x}] = 0$ and $\mathbb{D}_\mathcal{T}[\mathbf{x}[i]] = 1$ for any $i = 0, 1$. For both dimension, it is clear that $\mathbb{E}_\mathcal{T}[\mathbf{x}] = 0$ as the labels are balanced, and $\mathbb{D}_\mathcal{T}[\mathbf{x}[i]] = \frac{1}{2}\mu_t^2 + \sigma_t^2$ for any $i = 0, 1$ as $y\mathbf{x}[i] \sim N(\frac{\mu_t}{\sqrt{2}}, \sigma_t^2), (\mathbf{x}, y) \in \mathcal{T}$. Then we have $\frac{1}{2}\mu_t^2 + \sigma_t^2 = 1$. The same holds for the validation data such that $\frac{1}{2}\mu_v^2 + \sigma_v^2 = 1$.

Criterion 1 *The search procedure stops when there are two or more than two* skip-connect*s in one normal cell.*

The major advantage of the proposed stopping criterion is its simplicity. Compared with other DARTS variants, DARTS+ only needs a few modifications based on DARTS, and can significantly increase the performance with less search time. As too many *skip-connect*s will hurt the performance of DARTS, while an appropriate number of *skip-connect*s helps to transfer the information from the first layers to the last layers and stabilizing the training process, e.g., ResNet [10], which makes the architectures achieve better performance. Therefore, stopping by Criterion 1 is a reasonable choice.

The hyper-parameter *two* in Criterion 1 is motivated by P-DARTS [3], where the number of *skip-connect*s in the cell of final architecture is manually cut down to two. However, DARTS+ is essentially different from P-DARTS in dealing with the *skip-connect*s. P-DARTS does not intervene the number of *skip-connect*s during the search procedure, but only replace the redundant *skip-connect*s with other operations as post-processing after the search procedure finishes. In contrast, our DARTS+ ends up with desired architectures with a proper number of *skip-connect*s to avoid the collapse of DARTS. It controls the number of *skip-connect*s more directly and also more effectively (See Table 1 for a performance comparison between DARTS+ and P-DARTS).

Since the stable ranking of architecture parameters α for learnable operations indicates the saturated search procedure in DARTS, we can also use the following stopping criterion:

Criterion 2 *The search procedure stops when the ranking of architecture parameters α for learnable operations becomes stable for a determined number of epochs (e.g., 10 epochs).*

It can be seen from Fig. 2(i) that the saturated training point (stopping point with Criterion 2) is close to the stopping point when Criterion 1 holds (the red dash line in Fig. 2(i)(a)). We also remark that both criteria can be used freely as the stopping points are close. However, Criterion 1 is much easier to operate, but if one needs stopping more precisely or other search spaces are involved, Criterion 2 could be used instead. *Ten* epochs in Criterion 2 is a hyper-parameter and according to our experiments, when the ranking of operators remains the same for more than 6 epochs it can be considered stable, implying that this hyper-parameter is not sensitive and flexible to choose. We further remark that our early stopping paradigm solves an intrinsic issue of DARTS and is orthogonal to other tricks, thus it has the potential to be used in other DARTS-based algorithms to achieve better performance. Moreover, our method is very easy to complement than other methods like computing the eigenvalues of Hessian in validation loss [39].

We note that recent state-of-the-art differentiable architecture search methods also introduce the early stopping idea in an ad hoc manner. To avoid the collapse, P-DARTS [3] uses 1) searching for 25 epochs instead of 50 epochs, 2) adopting *dropout* after *skip-connect*s, and 3) manually reducing the number of *skip-connect*s to two. Auto-DeepLab [19] uses fewer epochs for searching the architecture parameters and finds that searching for more epochs does not bring benefits.

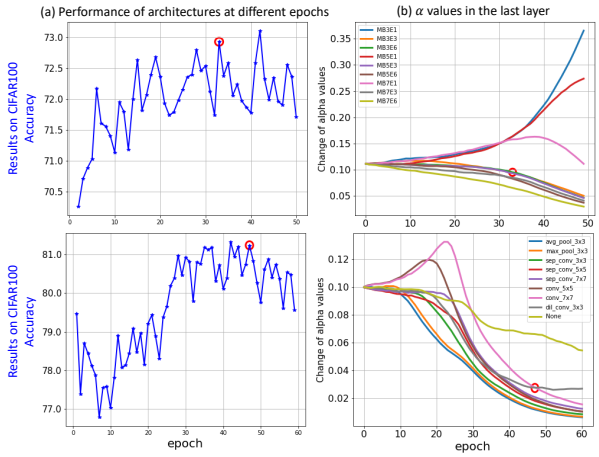


Figure 4: Results on the MobileNetV2 and ResNet search space. (a)(b) denote roughly the same as that in Figure 2(i). Red circle denotes the early stopping point at Criterion 2.

PC-DARTS [36] uses partial-channel connections to reduce search time, and therefore more epochs are needed for convergence of searching. Thus, setting 50 training epochs is also an implicit early stopping paradigm, see Appendix C.

4 Experiments and Analysis

4.1 Datasets

In this section, we conduct extensive experiments on benchmark classification datasets to evaluate the effectiveness of the proposed DARTS+ algorithm. We use four popular datasets including CIFAR10 [15], CIFAR100 [15], Tiny-ImageNet-200[5] and ImageNet [6]. CIFAR10/100 consists of 50K training images and 10K testing images and the resolution is 32×32. Tiny-ImageNet-200 contains 100K 64×64 training images and 10K testing images. ImageNet is obtained from ILSVRC2012 [28], which contains more than 1.2M training images and 50K validation images. We follow the general setting on the ImageNet dataset where the images are resized to 224×224 for training and testing.

4.2 Effectiveness of Early Stopping on Different Search Spaces

To verify the effectiveness of early stopping in DARTS+, we conduct extensive experiments with different datasets on selected architectures at different epochs. The experiments are carried out in two stages: architecture search and architecture evaluation.

DARTS Search Space. In the DARTS search space, the experimental settings are similar to DARTS. For CIFAR10 and CIFAR100, in the architecture search phase, we use the same one-shot model as the original DARTS and the hyperparameters are almost the same as DARTS except that a maximum of 60 epochs is adopted. In the architecture evaluation phase, the experimental settings follow the original

[5] https://tiny-ImageNet.herokuapp.com/

Table 1: Results of different architectures on CIFAR10 and CIFAR100. [1] denotes training without cutout augmentation. [2] denotes re-implementing with the proposed experimental settings. [3] denotes using a different search space from others. [4] denotes results of the best architecture searched from the corresponding dataset and training with more channels and more augmentations. * denotes using stopping Criterion 2, otherwise using Criterion 1.

Architecture	Search Dataset	Test Err. (%)		Param (M)	Search Cost (GPU days)	Search Method
		CIFAR10	CIFAR100			
DenseNet-BC [13][1]	-	3.46	17.18	25.6	-	manual
NASNet-A [42]	CIFAR10	2.65	-	3.3	1800	RL
AmoebaNet-B [27]	CIFAR10	2.55 ± 0.05	-	2.8	3150	evolution
PNAS [18][1]	CIFAR10	3.41 ± 0.09	-	3.2	225	SMBO
ENAS [26]	CIFAR10	2.89	-	4.6	0.5	RL
NAONet [22]	CIFAR10	3.18[1]	15.67	10.6	200	NAO
DARTS [21]	CIFAR10	3.00	17.76	3.3	1.5	gradient
DARTS[2]	CIFAR10	2.72	16.97	3.3	1.5	gradient
SNAS (moderate) [34]	CIFAR10	2.85	-	2.8	1.5	gradient
ProxylessNAS [2][3]	CIFAR10	2.08	-	5.7	4	gradient
P-DARTS [3]	CIFAR10	2.50	16.55	3.4	0.3	gradient
P-DARTS [3]	CIFAR100	2.62	15.92	3.6	0.3	gradient
ASAP [25]	CIFAR10	2.49 ± 0.04	15.6	2.5	0.2	gradient
PC-DARTS [36]	CIFAR10	2.57 ± 0.07	-	3.6	0.1	gradient
DARTS+	CIFAR10	$\mathbf{2.32}(2.50 \pm 0.11)$	16.28	3.7	0.4	gradient
DARTS+	CIFAR100	2.46	$\mathbf{14.87}(15.42 \pm 0.30)$	3.8	0.2	gradient
DARTS+*	CIFAR10	$\mathbf{2.20}(2.37 \pm 0.13)$	15.04	4.3	0.6	gradient
DARTS+*	CIFAR100	2.46	$\mathbf{14.87}(15.45 \pm 0.30)$	3.9	0.5	gradient
DARTS+ (Large)[4]	-	**1.68**	**13.03**	7.2	-	gradient

DARTS, except that 2000 epochs are used for better convergence. For Tiny-ImageNet-200, in the search phase, the one-shot model is almost the same as CIFAR10/100 except that a 3×3 convolution layer with stride 2 is added on the first layer to reduce the input resolution from 64×64 to 32×32. Other settings are the same as those used in CIFAR10/100.

We use Criterion 1 and 2 for early stopping in the DARTS search space. Other details of the experimental settings can be referred to in Appendix B.

The classification results of the selected architectures at different epochs are shown in Fig. 2(i). We also mark the "early stop" point under two criteria as "red dashed line" and "red circle" respectively. We observe that the selected architecture performs worse with larger epochs, implying that the original DARTS suffers from the collapse issue. In contrast, "early stopping" can generate good architectures at both stopping criteria, regardless of the type of datasets.

We also compare "early stopping" Criterion 1 and 2 in Table 1 and Fig. 2(i). We observe that both criteria achieve comparable performance on all datasets as the stopping points are very close.

MobileNetV2 and ResNet Search Space. To further verify the effectiveness of DARTS+, we use MobileNetV2 [29] and ResNet [10] as the backbone to build the architecture space [2]. For the MobileNetV2 search space, We introduce a set of mobile inverted bottleneck convolution (MBConv) with various kernel sizes and expansion ratios to construct the searching block. For ResNet search space, we construct the one-shot model by replacing the residual block with a set of candidate operations, where we keep the skip-connect in the residual block and involve 10 candidate operations. In both search spaces, softmax is applied to architecture parameters to compute the weights, which are used to determine the selected architecture. The experiments are conducted on CIFAR100 dataset. Details of the search spaces and experimental settings on architecture search and architecture evaluation are summarized in Appendix B.

Table 2: Results of different architectures on Tiny-ImageNet-200. † denotes directly searching on Tiny-ImageNet-200, otherwise transferred from CIFAR10. * denotes using Criterion 2, otherwise using Criterion 1

Architecture	Test Err. (%)	Params (M)
ResNet-18 [38]	47.3	11.7
DenseNet-BC [16]	37.1	-
NASNet	29.8	4.5
DARTS	30.4	3.8
DARTS†	46.1	2.1
SNAS	30.6	3.4
ASAP	30.0	3.3
DARTS+	**29.1**	4.2
DARTS+†	**28.3**	3.8
DARTS+*†	**27.6**	4.3

As the *skip-connect*s are not involved in the search spaces, we use "early stopping" Criterion 2. The classification results of the selected architectures at different epochs are shown in Fig. 4. The time to "early stop" with Criterion 2 is marked as "red circle". It can be seen that the selected architecture with "early stopping" achieves relatively the best performance, compared with randomly searched architecture (epoch 0) and that in large epochs.

Table 3: Results of different architectures on ImageNet. * denotes re-implementing the result. † denotes directly searching on ImageNet. ‡ denotes using SE-Module and training with more augmentations (AutoAugment, mixup, etc.)

Architecture	Test Err. (%)		Params	×+	Search Cost	Search Method
	Top-1	Top-5	(M)	(M)	(GPU days)	
MobileNet [11]	29.4	10.5	4.2	569	-	manual
MobileNet-V2 (1.4×) [29]	25.3	-	6.9	585	-	manual
ShuffleNet-V2 (2×) [23]	25.1	-	7.4	591	-	manual
NASNet-A [42]	26.0	8.4	5.3	564	1800	RL
AmoebaNet-C [27]	24.3	7.6	6.4	570	3150	RL
PNAS [18]	25.8	8.1	5.1	588	225	SMBO
MnasNet-92 [33]	25.2	8.0	4.4	388	-	RL
EfficientNet-B0 [32]	23.7	6.8	5.3	390	-	RL
DARTS [21]	26.7	8.7	4.7	574	4.0	gradient
SNAS (mild) [34]	27.3	9.2	4.3	522	1.5	gradient
ProxylessNAS [2]†	24.9	7.5	7.1	465	8.3	gradient
P-DARTS (CIFAR10) [3]	24.4	7.4	4.9	557	0.3	gradient
ASAP [25]	26.7	-	-	-	0.2	gradient
XNAS [24]	24.0	-	5.2	600	0.3	gradient
PC-DARTS [36]†	24.2	7.3	5.3	597	3.8	gradient
PC-DARTS*†	23.8	7.3	5.3	597	3.8	gradient
DARTS+ (CIFAR100)	**23.7**	**7.2**	5.1	591	0.2	gradient
DARTS+†	23.9	7.4	5.1	582	6.8	gradient
SE-DARTS+ (CIFAR100)‡	**22.5**	**6.4**	6.1	594	0.2	gradient

4.3 Comparison with State-of-the-Art

Unless specified, we use the DARTS search space and "early stopping" Criterion 1 evaluating DARTS+. Note that the stopping points by Criterion 1 and 2 are almost the same in the proposed search space, as discussed in Sec. 4.2.

For CIFAR10, CIFAR100, and Tiny-Imagenet-200 datasets, the experimental settings on both architecture search and architecture evaluation phase can be found in Sec. 4.2 and Appendix B. For ImageNet, following [36], the one-shot model starts with three 3×3 convolution layers with stride 2 to reduce the resolution from 224×224 to 28×28, and the rest of the network consists of 8 cells. We select 10% data from the training set for updating model weights, and another 10% for updating architecture parameters. In the architecture evaluation phase, we train the model for 800 epochs with batch size 2048 for better convergence. Other experimental settings are almost the same as those in DARTS, which can be found in Appendix B.

Search Results and Analysis. The proposed DARTS+ needs less searching time as "early stopping" is adopted. For CIFAR10, the search procedure requires 0.4 GPU days with a single Tesla V100 GPU and stops at about epoch 35. For CIFAR100, the searching time is 0.2 GPU days and the search procedure stops at about epoch 18. For Tiny-ImageNet-200, searching stops at about epoch 10. For ImageNet, the search procedure involves 200 epochs and requires 6.8 GPU days on Tesla P100 GPU.

The number of *skip-connect*s shown in Fig. 2(i) implies that the cells searched by DARTS+ contain a small number of *skip-connect*s, showing that DARTS+ succeeds in searching with all three datasets including CIFAR10/100, Tiny-ImageNet-200, and ImageNet. However, the original DARTS fails to search on CIFAR100 as the selected architecture is full of *skip-connect*s, and most previous works on differentiable search [3, 21, 34] do not search on ImageNet.

The selected architecture can be found in Appendix B.

Architecture Evaluation on CIFAR10 and CIFAR100. The evaluation results are summarized in Table 1. For each selected cell from either CIFAR10 or CIFAR100, we report the performance on both datasets. With the simple "early stopping" paradigm, we achieve the best results with 2.32% test error on CIFAR10 and 14.87% test error on CIFAR100. The proposed DARTS+ is much simpler and better than other modified DARTS algorithms like P-DARTS and PC-DARTS. ProxylessNAS uses a different search space, and it involves more search time. Moreover, DARTS+ is much easier to implement than other modified DARTS variants including ASAP.

We further increase the initial channel number from 36 to 50 and add more augmentation tricks including AutoAugment [5] and mixup [40] to achieve better results. Table 1 shows that DARTS+ achieves impressive 1.68% test error on CIFAR10 and 13.03% on CIFAR100, demonstrating the effectiveness of DARTS+.

Architecture Evaluation on Tiny-ImageNet-200. For DARTS and DARTS+, we use the architecture searched directly from Tiny-ImageNet-200 for evaluation. We also transfer the architectures searched from other algorithms for a fair comparison. The results are shown in Table 2. DARTS+ achieves the state-of-the-art 28.3% test error with Criterion 1 and 27.6% test error with Criterion 2. Note that architecture searched on Tiny-ImageNet-200 with DARTS has less parameter size and performs much inferior, because DARTS suffers from collapse and the architecture searched with DARTS contains lots of *skip-connect*s.

Architecture Evaluation on ImageNet. We use the architecture searched directly from ImageNet for evaluation, and the architecture from CIFAR100 to test the transferability of the selected architecture. The experimental results are shown in Table 3. Note that we re-implement PC-DARTS and the results are reported. When searching on ImageNet with the proposed DARTS+, the selected architecture achieves an impressive $23.9\%/7.4\%$ top-1/top-5 error, and the architecture transferred from CIFAR100 achieves state-of-the-art $23.7\%/7.2\%$ error. The results imply that DARTS with "early stopping" succeeds in searching for a good architecture with an impressive performance on large-scale datasets with limited time.

We also adopt SE-module [12] in the architecture transferred from CIFAR100, and introduce AutoAugment [5] and mixup [40] for training to obtain a better model. The results are shown in Table 3, and we achieve $22.5\%/6.4\%$ top-1/top-5 error with only additional 3M flops, showing the effectiveness of the selected architecture.

5 Conclusion

In this paper, we conduct a comprehensive analysis and extensive experiments to show that DARTS suffers from the collapse problem, which is mainly caused by the *overfitting* of the one-shot model in DARTS. We propose "DARTS+", in which the "early stopping" paradigm is introduced to avoid the collapse of DARTS. The experiments show that we succeed in searching on various benchmark datasets including large-scale ImageNet with limited GPU days, and the resulting architectures achieve state-of-the-art performance on all benchmark datasets. Moreover, the proposed "early stopping" criteria could be applied to various search spaces and many recent signs of progress of DARTS could use "early stopping" to achieve better results.

References

[1] Achille, A.; Rovere, M.; and Soatto, S. 2019. Critical learning periods in deep networks. *ICLR 2019*.

[2] Cai, H.; Zhu, L.; and Han, S. 2019. Proxylessnas: Direct neural architecture search on target task and hardware. In *7th International Conference on Learning Representations, ICLR*.

[3] Chen, X.; Xie, L.; Wu, J.; and Tian, Q. 2019. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *Proceedings of the IEEE International Conference on Computer Vision*, 1294–1303.

[4] Chu, X.; Zhou, T.; Zhang, B.; and Li, J. 2019. Fair darts: Eliminating unfair advantages in differentiable architecture search. *arXiv preprint arXiv:1911.12126*.

[5] Cubuk, E. D.; Zoph, B.; Mane, D.; Vasudevan, V.; and Le, Q. V. 2018. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*.

[6] Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *CVPR*.

[7] Frankle, J.; Schwab, D. J.; and Morcos, A. S. 2020. The early phase of neural network training. In *ICLR*.

[8] Gong, X.; Chang, S.; Jiang, Y.; and Wang, Z. 2019. Autogan: Neural architecture search for generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 3224–3234.

[9] Goyal, P.; Dollár, P.; Girshick, R.; Noordhuis, P.; Wesolowski, L.; Kyrola, A.; Tulloch, A.; Jia, Y.; and He, K. 2017. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*.

[10] He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*.

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

[12] Hu, J.; Shen, L.; and Sun, G. 2018. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 7132–7141.

[13] Huang, G.; Liu, Z.; Van Der Maaten, L.; and Weinberger, K. Q. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4700–4708.

[14] Kingma, D. P., and Ba, J. 2015. Adam: A method for stochastic optimization. In Bengio, Y., and LeCun, Y., eds., *3rd International Conference on Learning Representations, ICLR*.

[15] Krizhevsky, A.; Nair, V.; and Hinton, G. 2009. Cifar-10 and cifar-100 datasets. *URl: https://www. cs. toronto. edu/kriz/cifar. html* 6.

[16] Lan, X.; Zhu, X.; and Gong, S. 2018. Self-referenced deep learning. In *Asian Conference on Computer Vision*, 284–300. Springer.

[17] Li, G.; Zhang, X.; Wang, Z.; Li, Z.; and Zhang, T. 2019. Stacnas: Towards stable and consistent optimization for differentiable neural architecture search. *arXiv preprint arXiv:1909.11926*.

[18] Liu, C.; Zoph, B.; Neumann, M.; Shlens, J.; Hua, W.; Li, L.-J.; Fei-Fei, L.; Yuille, A.; Huang, J.; and Murphy, K. 2018. Progressive neural architecture search. In *ECCV*.

[19] Liu, C.; Chen, L.-C.; Schroff, F.; Adam, H.; Hua, W.; Yuille, A. L.; and Fei-Fei, L. 2019. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *CVPR*.

[20] Liu, B.; Zhu, C.; Li, G.; Zhang, W.; Lai, J.; Tang, R.; He, X.; Li, Z.; and Yu, Y. 2020. Autofis: Automatic feature interaction selection in factorization models for click-through rate prediction. *KDD*.

[21] Liu, H.; Simonyan, K.; and Yang, Y. 2019. DARTS: Differentiable architecture search. In *ICLR*.

[22] Luo, R.; Tian, F.; Qin, T.; Chen, E.; and Liu, T.-Y. 2018. Neural architecture optimization. In *NeurIPS*.

[23] Ma, N.; Zhang, X.; Zheng, H.-T.; and Sun, J. 2018. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 116–131.

[24] Nayman, N.; Noy, A.; Ridnik, T.; Friedman, I.; Jin, R.; and Zelnik-Manor, L. 2019. XNAS: neural architecture search with expert advice. In *Advances in Neural Information Processing System, NeurIPS*, 1975–1985.

[25] Noy, A.; Nayman, N.; Ridnik, T.; Zamir, N.; Doveh, S.; Friedman, I.; Giryes, R.; and Zelnik, L. 2020. Asap: Architecture search, anneal and prune. In *International Conference on Artificial Intelligence and Statistics*, 493–503. PMLR.

[26] Pham, H.; Guan, M. Y.; Zoph, B.; Le, Q. V.; and Dean, J. 2018. Efficient neural architecture search via parameter sharing. In *Proceedings of the 35th International Conference on Machine Learning, ICML*, volume 80 of *Proceedings of Machine Learning Research*, 4092–4101. PMLR.

[27] Real, E.; Aggarwal, A.; Huang, Y.; and Le, Q. V. 2019. Regularized evolution for image classifier architecture search. In *AAAI*.

[28] Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision* 115(3).

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

[30] Stamoulis, D.; Ding, R.; Wang, D.; Lymberopoulos, D.; Priyantha, B.; Liu, J.; and Marculescu, D. 2019. Single-path nas: Designing hardware-efficient convnets in less than 4 hours. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 481–497. Springer.

[31] Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2016. Rethinking the inception architecture for computer vision. In *CVPR*.

[32] Tan, M., and Le, Q. V. 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. In Chaudhuri, K., and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning, ICML*, 6105–6114.

[33] Tan, M.; Chen, B.; Pang, R.; Vasudevan, V.; Sandler, M.; Howard, A.; and Le, Q. V. 2019. Mnasnet: Platform-aware neural architecture search for mobile. In *CVPR*.

[34] Xie, S.; Zheng, H.; Liu, C.; and Lin, L. 2019. SNAS: stochastic neural architecture search. In *7th International Conference on Learning Representations, ICLR*.

[35] Xu, H.; Yao, L.; Zhang, W.; Liang, X.; and Li, Z. 2019a. Auto-fpn: Automatic network architecture adaptation for object detection beyond classification. In *ICCV*.

[36] Xu, Y.; Xie, L.; Zhang, X.; Chen, X.; Qi, G.-J.; Tian, Q.; and Xiong, H. 2019b. Pc-darts: Partial channel connections for memory-efficient differentiable architecture search. *arXiv preprint arXiv:1907.05737*.

[37] Yang, A.; Esperança, P. M.; and Carlucci, F. M. 2020. Nas evaluation is frustratingly hard. *ICLR 2020*.

[38] Yao, Q.; Xu, J.; Tu, W.-W.; and Zhu, Z. 2019. Differentiable neural architecture search via proximal iterations. *arXiv preprint arXiv:1905.13577*.

[39] Zela, A.; Elsken, T.; Saikia, T.; Marrakchi, Y.; Brox, T.; and Hutter, F. 2020. Understanding and robustifying differentiable architecture search. *ICLR*.

[40] Zhang, H.; Cissé, M.; Dauphin, Y. N.; and Lopez-Paz, D. 2018. mixup: Beyond empirical risk minimization. In *6th International Conference on Learning Representations, ICLR*.

[41] Zhou, P.; Xiong, C.; Socher, R.; and Hoi, S. C. 2020. Theory-inspired path-regularized differential network architecture search. *Advances in Neural Information Processing System, NeurIPS*.

[42] Zoph, B., and Le, Q. V. 2017. Neural architecture search with reinforcement learning. In *5th International Conference on Learning Representations, ICLR*.

[43] Zoph, B.; Vasudevan, V.; Shlens, J.; and Le, Q. V. 2018. Learning transferable architectures for scalable image recognition. In *CVPR*.

A Proof of Lemma 1

Proof. **P1**: denote $\alpha_1 = 1-\alpha_0$ and $\mathbf{W}_\alpha = \alpha_0\mathbf{I}+\alpha_1\mathbf{W}$, the gradients of $\mathbf{W}, \mathbf{w}_r$ regarding to the training loss $\mathcal{L}_{train}$ is

$$\nabla_{\mathbf{w}_r}\mathcal{L}_{train} = \sum_{(\mathbf{x}_i,y_i)\in\mathcal{T}} [\sigma(y_i\mathbf{w}_r^\top\mathbf{W}_\alpha\mathbf{x}_i)-1]y_i\mathbf{W}_\alpha\mathbf{x}_i,$$
$$\nabla_{\mathbf{W}}\mathcal{L}_{train} = \sum_{(\mathbf{x}_i,y_i)\in\mathcal{T}} [\sigma(y_i\mathbf{w}_r^\top\mathbf{W}_\alpha\mathbf{x}_i)-1]y_i\alpha_1\mathbf{w}_r\mathbf{x}_i^\top,$$

where $\sigma(x) = \frac{1}{1+\exp(-x)}$ is the sigmoid function.

Suppose that $y\mathbf{x} = \mu_t\mathbf{e}+\sigma_t\epsilon, (\mathbf{x},y)\in\mathcal{T}, \epsilon\sim N(0,\mathbf{I})$. Denote $\mathbf{v} = \mathbf{W}_\alpha^\top\mathbf{w}_r$ and $\mathbf{v}_\perp$ holds such that $\|\mathbf{v}_\perp\| = \|\mathbf{v}\|$ and $\mathbf{v}_\perp^\top\mathbf{v} = 0$. As ϵ is isotropic, ϵ can be written as $\sigma_t\epsilon = \sigma_t'(\mathbf{v}\epsilon_0+\mathbf{v}_\perp\epsilon_1), \epsilon_0,\epsilon_1\sim N(0,1)$ where $\sigma_t' = \sigma_t/\|\mathbf{v}\|$. Note that the training error is expected to be small, which corresponds to smaller σ_t. Then the gradients of $\mathbf{w}_r$ can be rewritten as

$$\begin{aligned}\nabla_{\mathbf{w}_r}\mathcal{L}_{train} =& \mathbb{E}_{\epsilon_0,\epsilon_1}\big\{[\sigma(\mu_t\mathbf{v}^\top\mathbf{e}+\sigma_t'\mathbf{v}^\top(\mathbf{v}\epsilon_0+\mathbf{v}_\perp\epsilon_1))-1]\\ &[\mu_t\mathbf{W}_\alpha\mathbf{e}+\sigma_t'\mathbf{W}_\alpha(\mathbf{v}\epsilon_0+\mathbf{v}_\perp\epsilon_1)]\big\}\\ =&\mu_t\mathbb{E}_{\epsilon_0}\big[\sigma(\mu_t\mathbf{v}^\top\mathbf{e}+\sigma_t'\|\mathbf{v}\|^2\epsilon_0)-1\big](\mathbf{W}_\alpha\mathbf{e})\\ &+\sigma_t'\mathbb{E}_{\epsilon_0,\epsilon_1}\big\{[\sigma(\mu_t\mathbf{v}^\top\mathbf{e}+\sigma_t\|\mathbf{v}\|\epsilon_0)-1]\epsilon_0\\ &\cdot(\mathbf{W}_\alpha\mathbf{v}+\mathbf{W}_\alpha\mathbf{v}_\perp\epsilon_1)\big\}\\ =&\lambda_1\mathbf{W}_\alpha\mathbf{e}+\lambda_2\mathbf{W}_\alpha\mathbf{v}.\end{aligned}$$

The last equality holds such that $\lambda_1 = \mu_t\mathbb{E}_{\epsilon_0}\big[\sigma(\mu_t\mathbf{v}^\top\mathbf{e}+\sigma_t\|\mathbf{v}\|\epsilon_0)-1\big], \lambda_2 = \sigma_t'\mathbb{E}_{\epsilon_0}\big\{[\sigma(\mu_t\mathbf{v}^\top\mathbf{e}+\sigma_t\|\mathbf{v}\|\epsilon_0)-1]\epsilon_0\big\}$. Consider σ_t is small underlying the training data and $\|\mathbf{v}\|$ is limited as $\{\mathbf{W}\mathbf{x},\mathbf{x}\in\mathcal{T}\}$ is normalized, it is expected that $|\lambda_1|\gg|\lambda_2|$. Thus we have $\nabla_{\mathbf{w}_r}\mathcal{L}_{train}\approx\lambda_1\mathbf{W}_\alpha\mathbf{e}$. If $\mathbf{w}_r$ is optimized with $\mathbf{W}$ fixed, we have $\mathbf{w}_r\propto\mathbf{W}_\alpha\mathbf{e}$ with gradient descent. $\propto$ denotes one vector/matrix is parallel to the other, where $\mathbf{a}\propto\mathbf{b}$ implies there exists $\psi\neq0$ such that $\mathbf{a}=\psi\mathbf{b}$.

The gradients of $\mathbf{W}$ is

$$\begin{aligned}\nabla_{\mathbf{W}}\mathcal{L}_{train} =&\mathbb{E}_{\epsilon_0,\epsilon_1}\big\{[\sigma(\mu_t\mathbf{v}^\top\mathbf{e}+\sigma_t'\mathbf{v}^\top(\mathbf{v}\epsilon_0+\mathbf{v}_\perp\epsilon_1))-1]\\ &\alpha_1[\mu_t\mathbf{w}_r\mathbf{e}^\top+\sigma_t'\mathbf{w}_r(\mathbf{v}\epsilon_0+\mathbf{v}_\perp\epsilon_1)]^\top\big\}\\ =&\alpha_1[\lambda_1\mathbf{w}_r\mathbf{e}^\top+\lambda_2\mathbf{w}_r\mathbf{v}^\top].\end{aligned}$$

Similar as above, it is expected that $|\lambda_1|\gg|\lambda_2|$ and therefore $\nabla_{\mathbf{W}}\mathcal{L}_{train}\approx\lambda_1\mathbf{w}_r\mathbf{e}^\top$, thus $\mathbf{W}\propto\alpha_1\mathbf{w}_r\mathbf{e}^\top$ with gradient descent.

Then we have $\mathbf{w}_r\propto\mathbf{W}_\alpha\mathbf{e} = \alpha_0\mathbf{e}+\alpha_1\mathbf{W}\mathbf{e} = \alpha_0\mathbf{e}+\gamma_1\mathbf{w}_r\mathbf{e}^\top\mathbf{e} = \gamma_0\mathbf{e}+\gamma_1\mathbf{w}_r$ where γ_0,γ_1 are certain constants. Thus during the iteration, it is expected that $\mathbf{w}_r\propto\mathbf{e}$ and therefore $\mathbf{W}\to\mathbf{W}^*\propto\mathbf{e}\mathbf{e}^\top$.

As $\|\mathbf{w}_r\| = r$, we have $\mathbf{w}_r = r\mathbf{e}$.

Denote $\mathbf{W}^* = \eta\mathbf{e}\mathbf{e}^\top = \frac{\eta}{2}\mathbf{1}_{2\times2}$. As $\{\mathbf{W}\mathbf{x},\mathbf{x}\in\mathcal{T}\}$ is normalized, the variance of first dimension is $\mathbb{D}_{\mathcal{T}}[(\mathbf{W}\mathbf{x})[0]] = \mathbb{D}_{\mathcal{T}}[\frac{\eta}{2}(\mathbf{x}[0]+\mathbf{x}[1])] = \frac{\eta^2}{4}\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{T}}[\mathbf{x}[0]^2+\mathbf{x}[1]^2+2(y\mathbf{x}[0])(y\mathbf{x}[1])] = \frac{\eta^2}{4}(1+1+2\cdot\frac{\sqrt{2}}{2}\mu_t\cdot\frac{\sqrt{2}}{2}\mu_t) = \frac{\eta^2(2+\mu_t^2)}{4}$ and so as $\mathbb{D}_{\mathcal{T}}[(\mathbf{W}\mathbf{x})[1]] = \frac{\eta^2(2+\mu_t^2)}{4}$. As the variance of each dimension should be 1, we have $\eta = \frac{2}{\sqrt{2+\mu_t^2}}$. Then $\mathbf{W}^*\mathbf{x} = (\eta\mathbf{e}^\top\mathbf{x})\mathbf{e}$, which completes the proof of **P1**.

P2. The gradient of α_0 regarding to $\mathcal{L}_{val}$ is

$$\frac{\mathrm{d}\mathcal{L}_{val}}{\mathrm{d}\alpha_0} = \sum_{(\mathbf{x}_i,y_i)\in\mathcal{V}}[\sigma(y_i\mathbf{w}_r^\top\mathbf{W}_\alpha\mathbf{x}_i)-1]y_i(\mathbf{w}_r^\top\mathbf{x}_i-\mathbf{w}_r^\top\mathbf{W}\mathbf{x}_i).$$

When **P1** holds, $\mathbf{w}_r = r\mathbf{e}, \mathbf{W} = \frac{2}{\sqrt{2+\mu_t^2}}\mathbf{e}\mathbf{e}^\top$, $\mathbf{W}_\alpha^\top\mathbf{w}_r = r(\alpha_0+\frac{2\alpha_1}{\sqrt{2+\mu_t^2}})\mathbf{e}$. Denote $\lambda = \alpha_0+\frac{2\alpha_1}{\sqrt{2+\mu_t^2}}$ and $\mathbf{x} = \mu_v\mathbf{e}+\sigma_v\epsilon, \epsilon\sim N(0,\mathbf{I})$, we have

$$\begin{aligned}\frac{\mathrm{d}\mathcal{L}_{val}}{\mathrm{d}\alpha_0} =& \sum_{(\mathbf{x}_i,y_i)\in\mathcal{V}}[\sigma(r\lambda y_i\mathbf{e}^\top\mathbf{x}_i)-1]ry_i(1-\lambda)\mathbf{e}^\top\mathbf{x}_i\\ =&\mathbb{E}_\epsilon\big\{[\sigma(r\lambda(\mu_v+\sigma_v\mathbf{e}^\top\epsilon))-1]r(1-\lambda)(\mu_v+\sigma_v\mathbf{e}^\top\epsilon)\big\}\\ =&r(1-\lambda)\mathbb{E}_{\epsilon_1}\big\{[\sigma(r\lambda(\mu_v+\sigma_v\epsilon_1))-1](\mu_v+\sigma_v\epsilon_1)\big\}.\end{aligned}$$

The last equality holds such that $\epsilon = \epsilon_1\mathbf{e}+\epsilon_2\mathbf{e}_\perp, \epsilon_1,\epsilon_2\sim N(0,1)$ where $\mathbf{e}_\perp = \frac{1}{\sqrt{2}}[-1,1]^\top$, in this case $\mathbf{e}^\top\epsilon = \epsilon_1$. Moreover, as $r>0, \mu_t\in[0,\sqrt{2}], 1-\lambda\le1-(\alpha_0+\alpha_1) = 0$ and the equality only holds when $\mu_t = \sqrt{2}$ or $\sigma_t = 0$.

Consider that $\frac{1}{2}\mu_v^2+\sigma_v^2 = 1$, we have $\mu_v\in[0,\sqrt{2}]$ and $\sigma_v\in[0,1]$. Denote

$$g(r,\sigma_v) = -\mathbb{E}_{\epsilon_1}\big\{[\sigma(r\lambda(\mu_v+\sigma_v\epsilon_1))-1](\mu_v+\sigma_v\epsilon_1)\big\}.$$

We will show there exists $\sigma_0(r)\in(0,1)$ such that $g(r,\sigma_v)<0, \sigma_v>\sigma_0(r)$. It is clear that $g(\sigma_v)$ is continuous at $[0,1]$. $g(r,0) = -\sqrt{2}[\sigma(\sqrt{2}r\lambda)-1]>0$. While $g(r,1) = -\mathbb{E}_{\epsilon_1}\big\{[\sigma(r\lambda\epsilon_1)-1]\epsilon_1\big\} = -\frac{1}{2}\mathbb{E}_{\epsilon_1}\big\{[\sigma(r\lambda\epsilon_1)-1]\epsilon_1\big\}-\frac{1}{2}\mathbb{E}_{\epsilon_1}\big\{[\sigma(-r\lambda\epsilon_1)-1](-\epsilon_1)\big\} = \frac{1}{2}\mathbb{E}_{\epsilon_1}\big\{[\sigma(-r\lambda\epsilon_1)-\sigma(r\lambda\epsilon_1)]\epsilon_1\big\}<0$ as $[\sigma(-r\lambda\epsilon_1)-\sigma(r\lambda\epsilon_1)]\epsilon_1\le0$ for any r,ϵ_1. As $g(r,1)<0<g(r,0)$ and $g(r,\sigma_v)$ is continuous for any $r>0$, by the Intermediate Value Theorem for continuous function, there exists $\sigma_0(r)\in(0,1)$ such that $g(r,\sigma_0(r)) = 0$ and $g(r,\sigma_v)<0$ if $\sigma_v>\sigma_0(r)$. Thus $\frac{\mathrm{d}\mathcal{L}_{val}}{\mathrm{d}\alpha_0}<0$ if $\sigma_v>\sigma_0(r)$ as $1-\lambda<0$, which means that α_0 will get larger with gradient descent.

Moreover, The partial derivative of $g(r,\sigma_v)$ regarding to r is

$$\begin{aligned}\frac{\partial g(r,\sigma_v)}{\partial r} =& -\mathbb{E}_{\epsilon_1}\big\{\frac{\partial}{\partial r}[\sigma(r\lambda(\mu_v+\sigma_v\epsilon_1))-1](\mu_v+\sigma_v\epsilon_1)\big\}\\ =&-\mathbb{E}_{\epsilon_1}\big\{\sigma(\cdot)(1-\sigma(\cdot))(\mu_v+\sigma_v\epsilon_1)^2\big\}\\ <&0.\end{aligned}$$

where $\sigma(\cdot)$ denotes $\sigma(r\lambda(\mu_v+\sigma_v\epsilon_1))$. It is clear that $g(r,\sigma_v)$ is momentously decreasing for any $\sigma_v\in[0,1]$. Consider $r_1,r_2>0$, if there exists $\sigma_0(r_1)$ such that $g(r_1,\sigma_0(r_1)) = 0$ and $g(r_1,\sigma_v)<0$ if $\sigma_v>\sigma_0(r_1)$, it is clear that $g(r_2,\sigma_0(r_1))<0$. With the Intermediate Value Theorem, there exists $\sigma_0(r_2)\in(0,\sigma_0(r_1))$ such that $g(r_2,\sigma_0(r_2)) = 0$ and $g(r_2,\sigma_v)<0$ if $\sigma_v>\sigma_0(r_2)$. Therefore, $\sigma_0(r_2)<\sigma_0(r_1)$ holds for any $r_2>r_1>0$, thus $\sigma_0(r)$ is a momentously decreasing function. Then the proof is completed.

Remark Fig. 5 shows the numerical value of $\sigma_0(r)$ when $\alpha_0 = 0.5$. In fact, $\sigma_0(r)$ is not sensitive to α_0.

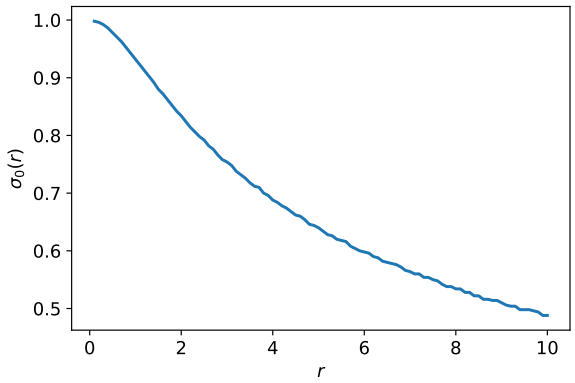


Figure 5: The numerical value of $\sigma_0(r)$.

B Basic Experimental Settings

B.1 Architecture Search

DARTS Search Space. Our search space on DARTS is the same as original DARTS [21] which has 8 candidate operations including *skip-connect, max-pool-3x3, avg-pool-3x3, sep-conv-3x3, sep-conv-5x5, dil-conv-3x3, dil-conv-5x5, zero*, and the structure of each operation is exactly the same as DARTS.

For CIFAR10 and CIFAR100, we use the same one-shot model as the original DARTS in which 8 cells (i.e. 6 normal cells and 2 reduction cells) with 16 channels are trained for searching. We use half of the training data to train the model weights and the other half to update the architecture parameters. We search for a maximum of 60 epochs with batch size 64. SGD is used to optimize the model weights with initial learning rate 0.025, momentum 0.9, and weight decay 3×10^{-4}. Adam [14] is adopted to optimize architecture parameters with initial learning rate 3×10^{-4}, momentum $(0.5, 0.999)$ and weight decay 10^{-3}. Early stopping is applied at a certain epoch when Criterion 1 is met.

For Tiny-ImageNet-200, the one-shot model is almost the same as CIFAR10/100 except that a 3×3 convolution layer with stride 2 is added on the first layer to reduce the input resolution from 64×64 to 32×32. Other settings are the same as those used in CIFAR10/100, including the "early stopping" criterion.

For ImageNet, following [36], the one-shot model starts with three 3×3 convolution layers with stride 2 to reduce the resolution from 224×224 to 28×28, and the rest of the network consists of 8 cells. We select 10% data from the training set for updating model weights and another 10% for updating architecture parameters. We search with batch size 512 for both training and validation sets. SGD is used for model weights training with initial learning rate 0.2 (cosine learning rate decay), momentum 0.9, and weight decay 3×10^{-4}. The architecture parameters are trained with Adam with learning rate 3×10^{-3}, momentum $(0.5, 0.999)$ and weight decay 10^{-3}.

For all the datasets, the one-shot model weights and architecture parameters are optimized alternatively. The cell structure is determined by architecture parameters, following DARTS [21].

Table 4: The MobileNetV2 backbone model used for searching and evaluation. c, n, s are the same as that in [29].

Input	Operator	c	n	s
$32^2 \times 3$	conv2d	32	1	1
$32^2 \times 32$	bottleneck	16	1	1
$32^2 \times 16$	bottleneck	24	4	2
$16^2 \times 24$	bottleneck	40	4	2
$8^2 \times 40$	bottleneck	80	4	2
$4^2 \times 80$	bottleneck	96	4	1
$4^2 \times 96$	bottleneck	192	4	1
$4^2 \times 192$	bottleneck	320	1	1
$4^2 \times 320$	avgpool 4x4	-	1	-
$1^2 \times 320$	conv2d 1x1	k	1	-

The selected architectures are shown in Fig. 6. We observe that the cells searched by DARTS+ contain most *convolution*s and a few *skip-connect*s.

MobileNetV2 Search Space. In this search space, CIFAR100 is used for searching and evaluating. We follow [2] and use MobileNetV2 [29] as the backbone to construct architecture space. We initiate each layer of the one-shot model by a set of mobile inverted bottleneck convolution(MBConv) layers with various kernel sizes $\{3, 5, 7\}$, and expansion rations $\{3, 6\}$. As the input image size is just 32×32, The backbone is slightly different from the original MobielNetV2, shown in Fig. 4. Different from [2] which uses the binary gate to connect operations within each layer, we follow DARTS and apply softmax to architecture parameters to get weights of candidate operations. The architecture is determined by choosing operations with the largest architecture parameter α in each layer. We search the model on CIFAR100 for 50 epochs and perform early stopping with Criterion 2, which is based on the rank of last layer architecture parameters.

The selected architecture with early stopping Criterion 2 is shown in Fig. 7.

ResNet Search Space. Considering the success of ResNet [10] on various vision tasks, we use ResNet-50 as the backbone to develop the architecture search space. As the input size is 32×32, the architecture is modified such that the first layer is 3x3 convolution with stride 1, and the first max-pool layer is removed, so that the output feature map size before the global average pooling layer is 4×4. We construct the one-shot model by replacing *conv-3x3* in each residual block with a search cell. Similar to the inception module [31], a search cell is composed of 4 branches of candidate operations. The goal is to choose one operation per branch by applying the softmax operation within the branch. The outputs of each branch are concatenated together as the final output of the search cell. Note that every search branch involves 10 candidate operations including *zero, max-pool-3x3, avg-pool-3x3, sep-conv-3x3, sep-conv-5x5, sep-conv-7x7, dil-conv-3x3, conv-3x3, conv-5x5, conv-7x7*. Softmax is applied to architecture parameters to compute the weights, which are used to determine the selected architecture. We search the model on CIFAR100 for at most

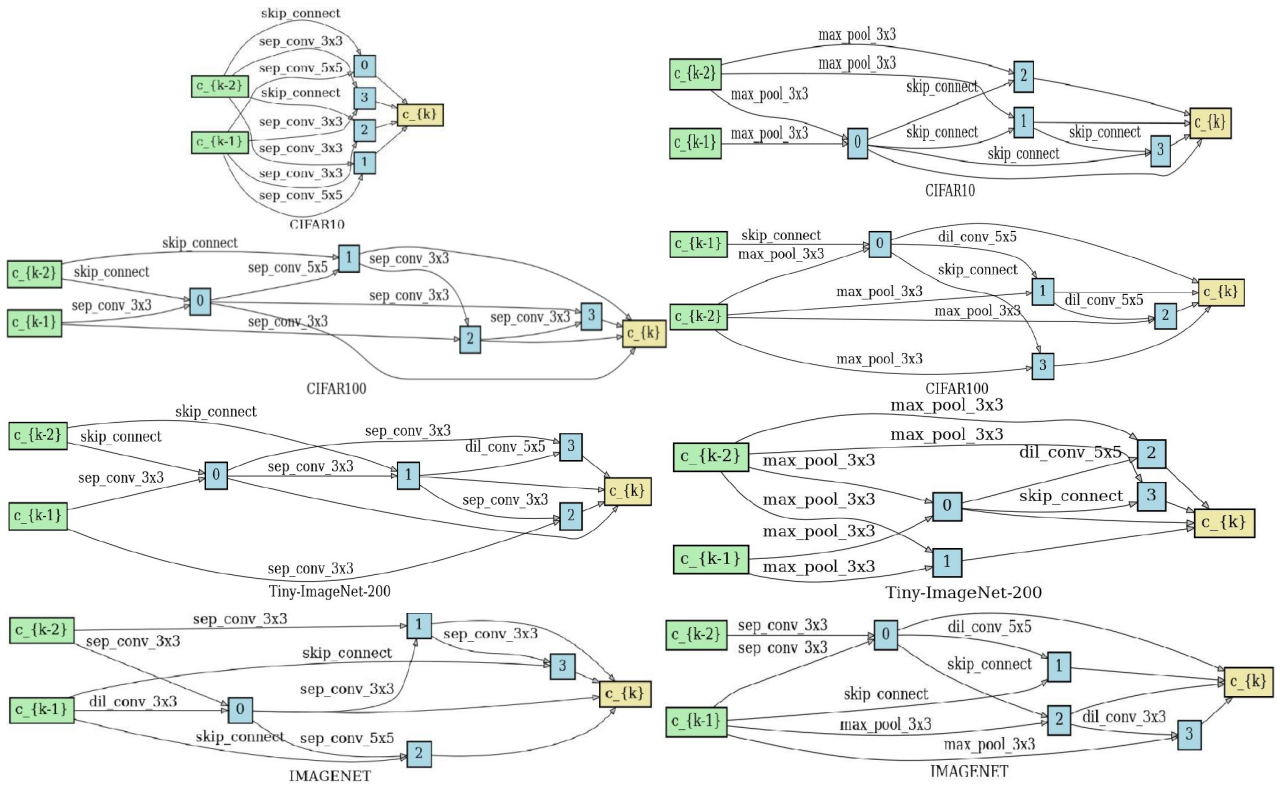


Figure 6: The cell of best structures searched on different datasets.

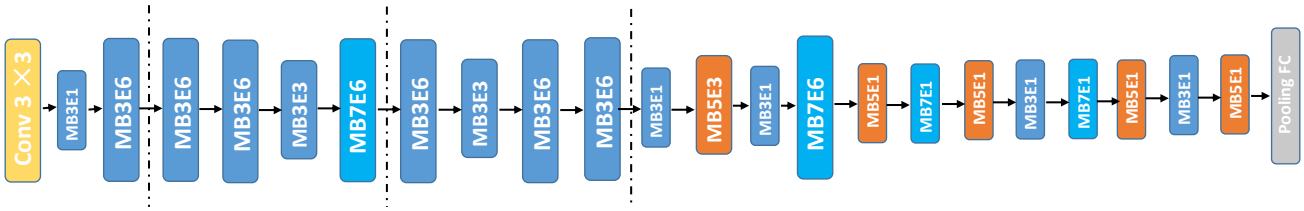


Figure 7: Architecture selected by DARTS+ in MobileNet-V2 search space on CIFAR100.

80 epochs and use Criterion 2 for early stopping based on the ranking of the last layer's architecture parameters.

The selected architecture with early stopping Criterion 2 is shown in Fig. 8.

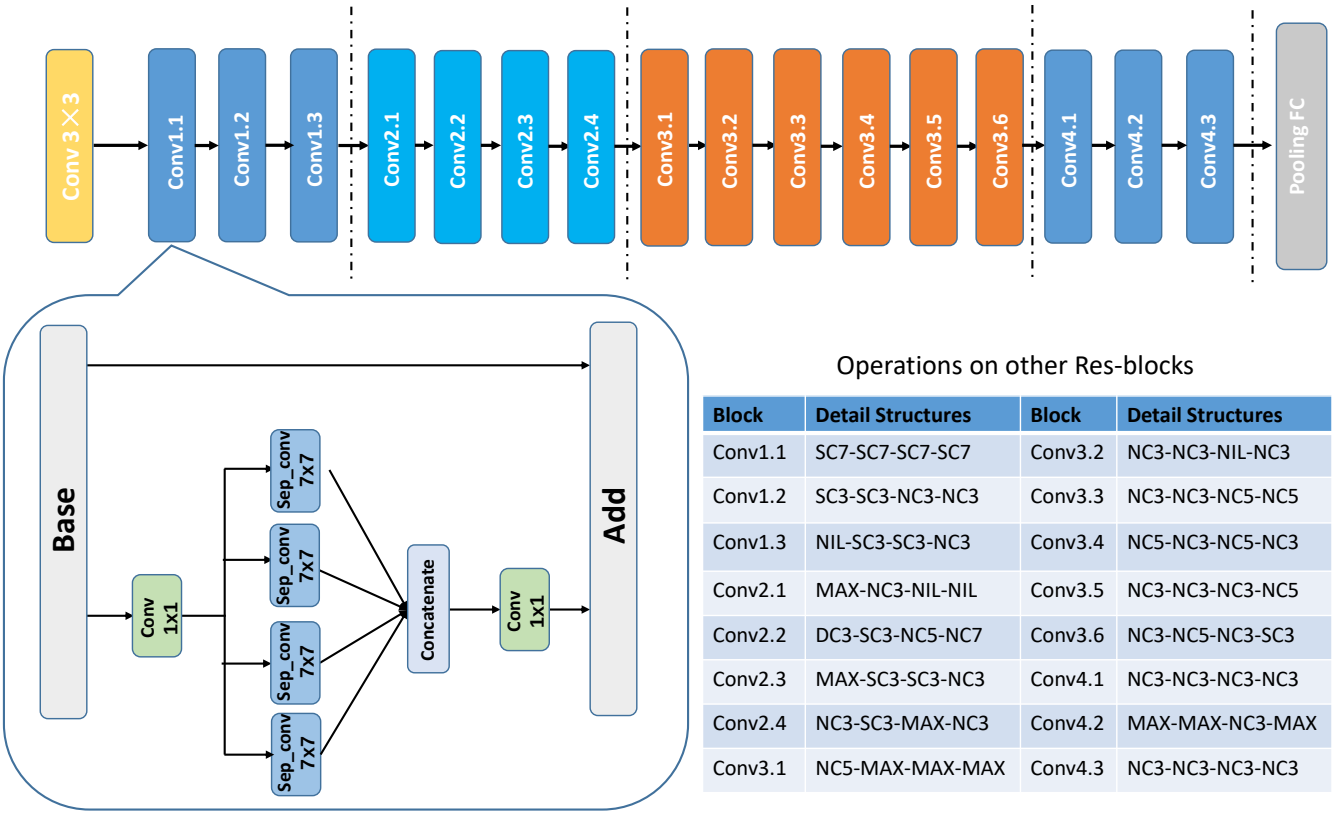
B.2 Architecture Evaluation

DARTS Search Space. For each selected architecture, we follow the configurations and hyper-parameters of the previous works [3, 21] for evaluation on different datasets.

When evaluating CIFAR10 and CIFAR100, we use a network of 20 cells and 36 initial channels for evaluation to ensure a comparable model size as other baseline models. We use the whole training set to train the model for 2000 epochs with batch size 96 to ensure convergence. Other hyper-parameters are set the same as the ones in the search stage. Following existing works [18, 26, 27, 43], we also add some enhancements including cutout, path dropout with probability 0.2 and auxiliary towers with weight 0.4. We further increase the initial channel number from 36 to 50 and add more augmentation tricks including AutoAugment [5] and mixup [40] to achieve better results. All the results are shown in Table 1 in the main paper.

For Tiny-ImageNet-200, the network is similar to CIFAR10/100 where 20 cells and 36 channels are involved, except that an additional 3×3 convolution layer with stride 2 is inserted in the first layer. We also transfer the architectures searched from other algorithms to Tiny-ImageNet-200 and evaluate the performance for a fair comparison. Other experimental settings are the same as CIFAR10/100. The results are shown in Table 2 in the main paper.

For ImageNet, We use the architecture searched directly from ImageNet for evaluation, and the architecture from CI-



Operations on other Res-blocks

Block	Detail Structures	Block	Detail Structures
Conv1.1	SC7-SC7-SC7-SC7	Conv3.2	NC3-NC3-NIL-NC3
Conv1.2	SC3-SC3-NC3-NC3	Conv3.3	NC3-NC3-NC5-NC5
Conv1.3	NIL-SC3-SC3-NC3	Conv3.4	NC5-NC3-NC5-NC3
Conv2.1	MAX-NC3-NIL-NIL	Conv3.5	NC3-NC3-NC3-NC5
Conv2.2	DC3-SC3-NC5-NC7	Conv3.6	NC3-NC5-NC3-SC3
Conv2.3	MAX-SC3-SC3-NC3	Conv4.1	NC3-NC3-NC3-NC3
Conv2.4	NC3-SC3-MAX-NC3	Conv4.2	MAX-MAX-NC3-MAX
Conv3.1	NC5-MAX-MAX-MAX	Conv4.3	NC3-NC3-NC3-NC3

Figure 8: Architecture selected by DARTS+ in ResNet based search space on CIFAR100. "SC" denotes separable convolution, "NC" denotes normal convolution, "MAX" denotes max pooling layer, "NIL" denotes zero operation, "DC" denotes dilated convolution.

FAR100 to test the transferability of the selected architecture. We follow DARTS such that the number of cells is 14, and the initial number of channels is 48. We train the model for 800 epochs with batch size 2048 on 8 Nvidia Tesla V100 GPUs as more epochs can achieve better convergence. The model is optimized with the SGD optimizer with an initial learning rate 0.8 (cosine decayed to 0), the momentum of 0.9, and weight decay 3×10^{-5}. We use learning rate warmup [9] for the first 5 epochs and other training enhancements including label smoothing [31] and auxiliary loss tower. We also adopt SE-module [12] in the architecture transferred from CIFAR100, and introduce AutoAugment [5] and mixup [40] for training to obtain a better model. The experimental results are shown in Table 3 in the main paper.

MobileNetV2 and ResNet Search Space. Then, we follow almost the same configurations and hyper-parameters as used in DARTS [3, 21] to evaluate the selected architectures, except that the number of training epochs is 1,200 in MobileNetV2 and 600 in ResNet. To show the effectiveness of the selected architecture with early stopping, we train the architectures selected at other epochs for comparison. Fig. 4 in the main paper shows the effectiveness of the selected architecture with DARTS+, achieving 72.43% in the MobileNetV2 search space and 81.23% in the ResNet search space.

C Additional Experiments

C.1 More Illustrations on the Collapse of DARTS

To further show the collapse problem in DARTS and validate the effectiveness of the proposed criteria, we perform more experiments on CIFAR10, CIFAR100, and Tiny-Imagenet-200 datasets. The experimental settings are the same as those in the main paper. For each selected cell at different epochs, we conduct extensive experiments for evaluation. The experimental results are shown in Fig.9, 10 and 11. We can arrive at similar conclusions shown in Sec. 2.2 in the main paper and both criteria are shown in Sec. 3 select similar architectures and the architectures achieve comparable performance.

C.2 Implicit Early Stopping in PC-DARTS

We carry out experiments to show that some recent progress of DARTS such as PC-DARTS [36] adopts the early stopping idea implicitly. We use the original codes and settings of PC-DARTS [36] and searching on CIFAR100 for 600 epochs. The selected normal cells at 200/400/600 epochs in the searching procedure are shown in Fig.12. As the number

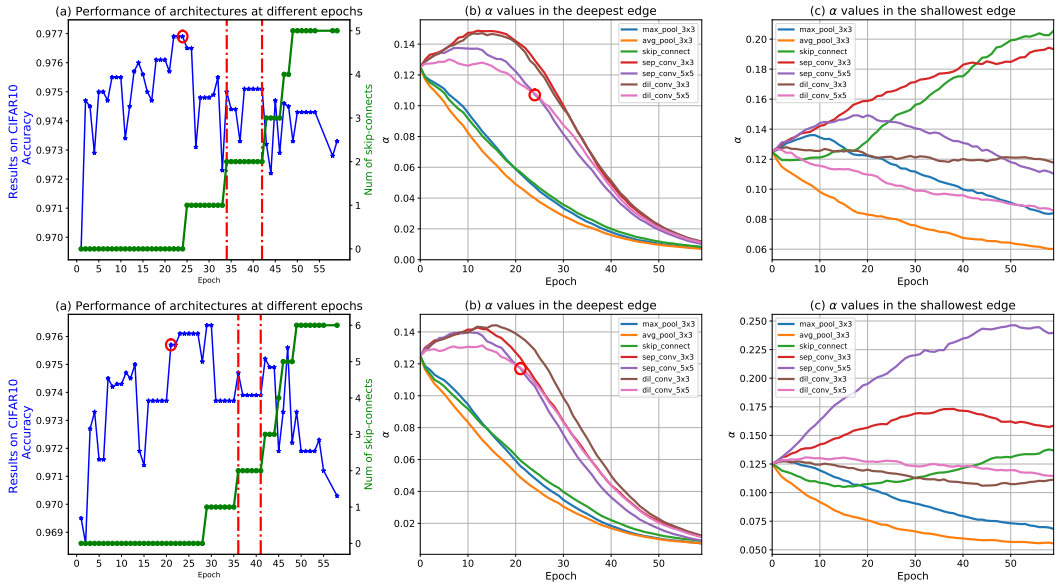


Figure 9: More results on CIFAR10 dataset. The meaning of each figure is the same as that in Fig. 2 in the main paper.

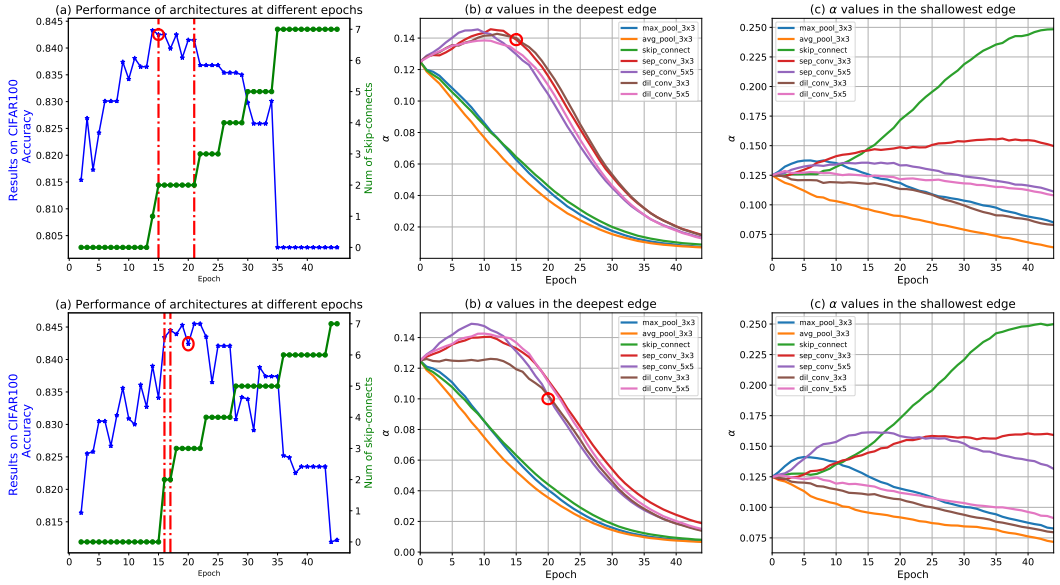


Figure 10: More results on CIFAR100 dataset. The meaning of each figure is the same as that in Fig. 2 in the main paper.

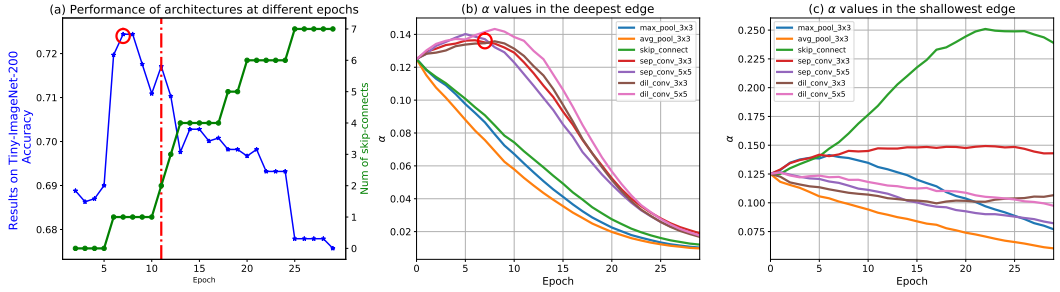


Figure 11: More results on Tiny-ImageNet-200 dataset. The meaning of each figure is the same as that in Fig. 2 in the main paper.

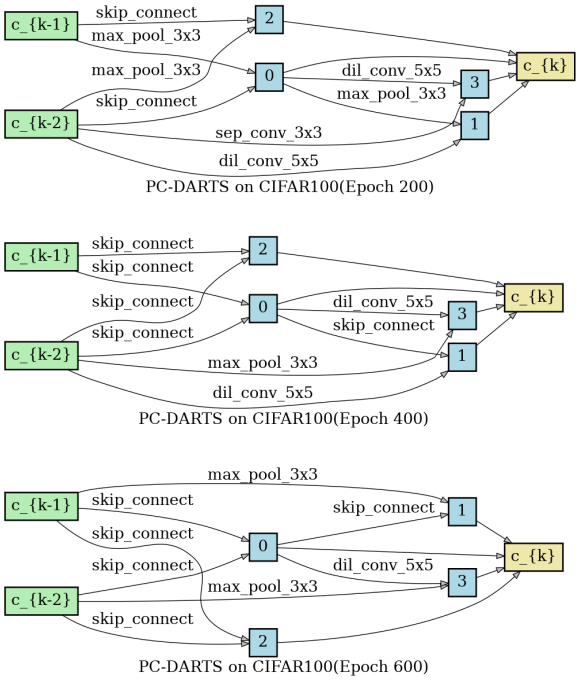


Figure 12: Selected normal cells at different epochs with PC-DARTS on CIFAR100.

of epoch increases, the number of skip-connects in the selected normal cells increases (2 skip-connects at 200 epochs and 5 at 600 epochs). It implies that PC-DARTS may suffer from collapse with more searching epochs, and training with just 50 epochs (used in DARTS and PC-DARTS) is an implicit early-stopping scheme to obtain better architecture.