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. In particular, a hyper-network called one-shot model is introduced, over which the architecture can be searched continuously with gradient descent. 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 exist cooperation and competition in the bi-level optimization in DARTS, where the architecture parameters and model weights are updated alternatively. 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 demonstrate that the proposed early stopping criterion is effective in avoiding the collapse issue. We also conduct experiments on benchmark datasets and show the effectiveness of our DARTS+ algorithm, where 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 Automatic Machine Learning (AutoML), which has attracted lots of attention recently. The neural architectures searched by NAS have achieved the state-of-the-art results over handcrafted neural architectures in various tasks, including object classification [4, 20, 21, 25, 26, 40], object detection [7, 36], semantic segmentation [19], recommender systems [14], etc.

The common practice of NAS first derives an architecture search space, and then finds the best architecture in the search space with a specific search method. Early works of NAS usually adopt the REINFORCE method [25, 40] and evolutionary algorithms [26] for searching effective architectures. However, obtaining state-of-the-art architectures with such methods involves huge computational cost (e.g., thousands of GPU days [40]), because a large number of architectures need to be trained and evaluated. Recently, one-shot methods [2, 3, 20, 25, 34] are proposed to reduce the search cost. Among the one-shot methods, Liu, Simonyan, and Yang [20] propose the Differentiable Architecture Search (DARTS) method to relax the search space to be continuous, so that one can search architectures and learn model weights directly with gradient descent. In particular, DARTS encodes the architecture search space with continuous parameters and performs searching with bi-level optimization, where the model weights and architecture parameters are optimized with training data and validation data alternatively. DARTS can reduce the search cost from thousands of GPU days to a few GPU days while keeping comparable performance.

Despite the efficiency of DARTS, a severe issue underlying DARTS has been found [4]. Namely, after a certain searching epochs, the number of *skip-connect*s increases dramatically in the selected architecture, which results in poor performance of the selected architecture. We call the phenomenon of performance drop after a certain number of epochs the "collapse" of DARTS. To tackle such issue, Chen et al. [4] propose a search space regularization in their P-DARTS, where dropout [29] is used to alleviate the dominance of *skip-connect*s during the search procedure, and the number of *skip-connect*s is manually controlled after the search procedure. However, this approach involves more hyper-parameters like dropout rate and the number of *skip-connect*s, which need to be carefully tuned by human experts. Moreover, Stamoulis et al. [30] propose Single-Path NAS using 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 space of Single-Path NAS is carefully designed. If searching is done in the original search space of DARTS, one-level optimization will perform worse than bi-level ones [20]. In summary, the mechanism of the collapse of DARTS is still unknown and needs to be addressed.

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

†Corresponding email: sunjiacheng1@huawei.com

In this paper, we first show the cause of the collapse of DARTS is that there exist *cooperation* and *competition* in the bi-level optimization in DARTS, where the architecture parameters and model weights are updated alternatively. In particular, we give an explanation of why lots of *skip-connect*s are involved in the selected architectures in DARTS and why they hurt the performance. To avoid the collapse of DARTS, we add the simple and effective "early stopping" paradigm to the original DARTS, named "DARTS+", where the search procedure stops by a certain criterion, illustrated in Fig. 1(a). We remark that recent progresses of DARTS, including P-DARTS [4], Auto-DeepLab [19] and PC-DARTS [37], also adopt the early stopping idea implicitly where fewer search epochs are manually set in their methods.

Moreover, we conduct sufficient experiments on benchmark datasets to demonstrate the effectiveness of the proposed DARTS+ algorithm. Specifically, DARTS+ achieves the state-of-the-art 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 the state-of-the-art 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 achieves 23.9% top-1 error.

In summary, our main contributions are listed as follows:

- We study the collapse issue of the DARTS method, and point out the underlying reason is the cooperation and competition in the bi-level optimization.
- We introduce an efficient "early stopping" paradigm to DARTS to avoid the collapse, and propose an effective criterion for early stopping.
- we conduct extensive experiments on benchmark datasets to demonstrate the effectiveness of the proposed algorithm, which achieves the state-of-the-art results on all of them.

2 Collapse of DARTS

There is a severe issue underlying DARTS [20], that lots of *skip-connect*s tend to appear in the selected architecture when the number of searching epoch 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. Moreover, we will discuss the cause of the collapse issue.

2.1 Preliminary: DARTS

The goal of DARTS is to search for a cell, which can be stacked to form a convolutional network or recursively connected to form a recurrent network. Each cell can be regraded as 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 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 [20].

2.2 Collapse Issue

It has been observed in DARTS that lots of *skip-connect*s 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 alpha value of *skip-connect*s (green line in Fig. 2(c)) becomes large when the number of search epochs is large, and thus the number of *skip-connect*s increases in the selected architecture as shown in the green line in Fig. 2(a). Such shallow network has less learnable parameters than deep ones, 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(a). To be more intuitive, we draw the selected architectures from different search epochs on CIFAR100 in Fig. 1(b). When the number of search epochs increases, the number of *skip-connect*s in the selected architecture also increases. Such phenomenon can also be observed on other datasets, such as CIFAR10 and ImageNet.

To avoid the collapse, one might propose to adjust search 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 would only delay the collapse as the choice of hyper-parameters is not the essential cause of collapse.

The underlying reason of the collapse issue is that there exist *cooperation* and *competition* in the bi-level optimization in DARTS, where the architecture parameters and model weights are updated alternatively. Intuitively, the architecture parameters and model weights are well optimized at the beginning, and then gradually turn to compete against each other after a while. Since the model weights have more

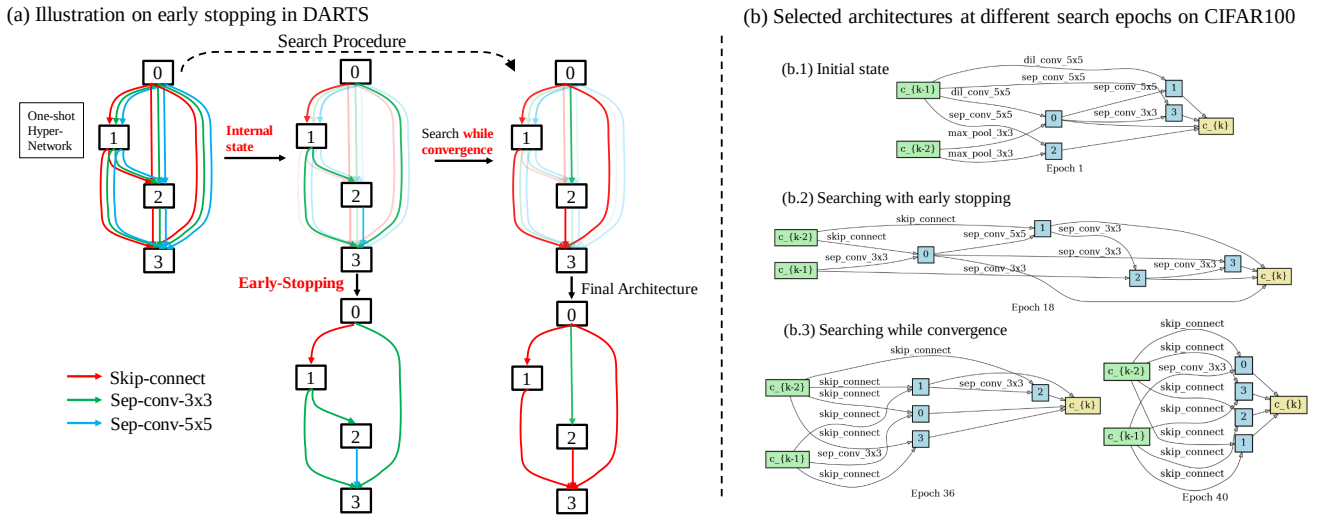

Figure 1: (a) Illustration of the early stopping paradigm. (b) Selected architectures at different search epochs on CIFAR100. With a proper early stopping mechanism, we may obtain better architectures, or we may end up with collapsed architectures with lots of skip-connects.

advantages than the architecture parameters in the competition, (e.g., the number of the model weights is far more than the number of architecture parameters, the architecture parameters are not sensitive to the final loss in the bi-level optimization, etc.), the architecture parameters cannot beat the model weights in the competition. As a result, the performance of selected architecture will first increase and then decrease (See blue lines in Fig. 2(a)).

In particular, at the initial state of the search procedure, the one-shot model underfits the training data. Thus, architecture parameters α and model weights w, which are learnable parameters of the one-shot model, will get better together at the beginning of the search procedure. This is the cooperation period. Note that the first cells in the whole one-shot model can touch the fresh data information, while the data that feed to the last cells are much noisier. If we allow different cells to have distinct architectures in the one-shot model, the first cells will learn features more quickly than the last cells. Since the feature representation learned by the last cells is relatively worse than the one learned by the first cells, the last cells are more likely to select more *skip-connect*s to obtain the good feature representation directly from the first cells.

Fig. 3 shows the learned normal cell architectures at different layers if we allow different architectures at different stages[1]. It can be seen that the algorithm tends to select deep architectures with learnable operations (namely, operations with parameters to be learned such as *convolutions*) in the first cells (Fig. 3(a)), while architectures with many *skip-connect*s are preferred in the last cells (Fig. 3(c)). If different cells are forced to have the same architecture, as DARTS does, *skip-connect*s will be broadcasted from the last cells to the first cells. With the increase of searching epochs, the number of *skip-connect*s in the selected architecture will largely grow. During this period, the architecture becomes bad, thus the competition of architecture parameters α and model weights w occurs, making the performance collapse.

Moreover, the cooperation and competition phenomenon can also be observed in other bi-level optimization problems (e.g., GAN, meta-learning, etc.). Take GAN as an example, it is proved that a good learned discriminator is essential for training the generator [8], which is the cooperation between generator and discriminator. However, if the input data (fake or real) lies in low-dimensional manifold and the discriminator is over-parameterized, the discriminator will easily separate the generated fake data from the real, and the generator will suffer from gradient vanishment and fail to generate real data [1], which is the competition.

3 The Early Stopping Methodology

Since the collapse issue of DARTS is caused by the cooperation and competition 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 a certain epoch, when DARTS starts to collapse. Such paradigm leads to both better performance and less search cost than the original DARTS. We remark that in this paper, we still follow the architecture sharing mechanism among cells used by DARTS[2]. We use DARTS+ to denote the DARTS algorithm with our early stopping criterion, which is stated as follows.

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

[1] Stages are split with reduction cells, and each stage consists of a number of stacked cells.

[2] In future work, we may relax this constraint and allow different cells or stages to have different structures.

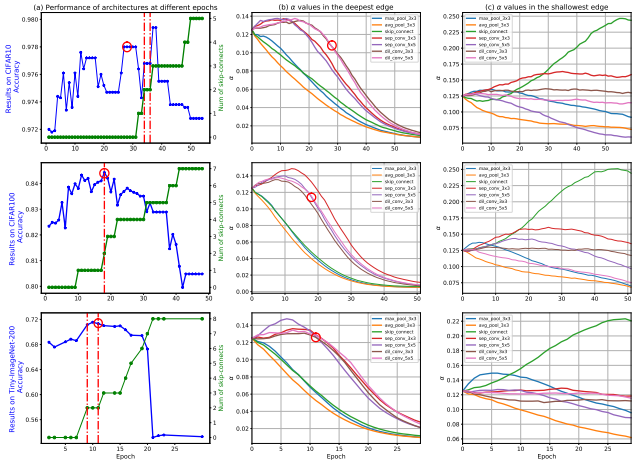

Figure 2: The collapse issue of DARTS. (a) The performance of architectures at different epochs on CIFAR10, CIFAR100 and Tiny-ImageNet-200, respectively (in blue line), and the number of *skip-connect*s in the normal cell (in green line). (b) The change of α in the deepest edge (connecting the last two nodes) of 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.

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 mentioned in Sec. 2.2, too many *skip-connect*s will hurt the performance of DARTS. On the other hand, an appropriate number of *skip-connect*s is helpful for transferring the information from first layers to 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.

Criterion 1 is motivated by P-DARTS [4], where the number of *skip-connect*s in the cell of final architecture is manually cut down to two. Although both DARTS+ and P-DARTS keep two *skip-connect*s in the cell of their final architectures, 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 replaces the redundant *skip-connect*s with other operations as a 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).

Now we give some intuition for Criterion 1 in Fig. 2. The red circles in Fig. 2(a-b) denote the points that the ranking of architecture parameters α for learnable operations (e.g., *convolution*s) becomes stable on CIFAR10, CIFAR100 and Tiny-ImageNet-200, respectively. Note that only the operation with the maximum α value is chosen in the selected architecture. When the ranking of α becomes stable, the operations to be selected are nearly determined, which implies that the search is almost saturated. The experiments also verify that after the point that the ranking of α becomes stable (red circles in Fig. 2(a)), the validation accuracies of selected architectures on all datasets (blue lines) tend to decrease, i.e., collapse. It can be seen that this point is close to the stopping point when Criterion 1 holds (the red dash line in Fig. 2(a)), thus stopping by Criterion 1 can prevent the collapse of DARTS.

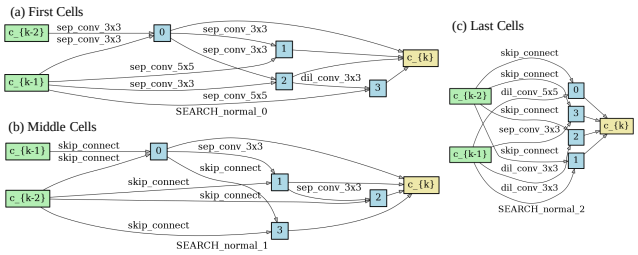

Figure 3: The selected architecture of normal cells at different layers when searching distinct cell architectures in different stages (stages are split with reduction cells). The searched dataset is CIFAR100. The first cells contain mostly convolutions, while the last cells are shallow with numerous *skip-connect*s.

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 1* *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).*

We remark that Criterion 1 is much easier to operate, but if one needs stopping more precisely or other search spaces are involved, Criterion 1* could be used instead. We further remark that our early stopping paradigm solves an intrinsic issue of DARTS and is orthogonal to other tricks, thus it has potential to be used in other DARTS-based algorithms to achieve better performance.

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 [4] 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] starts to update architecture parameters for a few epochs after updating weight parameters. PC-DARTS [37] 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.

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 [16], CIFAR100 [16], Tiny-ImageNet-200[3] 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 [27], 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 Architecture Search

Unless specified, we use Criterion 1 as the stopping condition for DARTS+ in the experiments. Note that the stopping points by Criterion 1 and 1* are almost the same in the proposed search space, and we will discuss the two criteria in detail in Sec. 4.4.

Implementation Details. We have similar experimental settings as DARTS. The experiments are carried out in two stages: architecture search and architecture evaluation. The search space is the same as DARTS 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 a 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. We use SGD to optimize the model weights with initial learning rate 0.025, momentum 0.9 and weight decay 3×10^{-4}. Adam [15] is used 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 certain epoch when Criterion 1 introduced in Sec. 3 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 [37], 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 [20].

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 35 epochs. For CIFAR100, the searching time is 0.2 GPU days and the search procedure stops at about 18 epochs. For Tiny-ImageNet-200, searching stops at about 10 epochs. For ImageNet, the search procedure involves 200 epochs, and it requires 6.8 GPU days on Tesla P100 GPU.

The selected architectures are shown in Fig. 4. We observe that the cells searched by DARTS+ contain most *convolution*s and a few *skip-connect*s.

It should be noticed 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 [4], and most previous works on differentiable search [4, 20, 34] do not search on ImageNet.

4.3 Architecture Evaluation

For each selected architecture, we follow the configurations and hyper-parameters of the previous works [4, 20] for evaluation on different datasets.

Results on CIFAR10 and CIFAR100. We use 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, 25, 26, 41], we also add some enhancements including cutout, path dropout with probability 0.2 and auxiliary towers with weight 0.4.

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 results are much better than the original DARTS, which gets 3% test error on CIFAR10 and 19.5% test error on CIFAR100 (See Fig. 2(a) with large epochs). 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 point out that "early stopping" can be used in many other differentiable search algorithms and search spaces to obtain better architectures.

We further increase the initial channel number from 36 to 50, and add more augmentation tricks including AutoAug-

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

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 [40]	CIFAR10	2.65	-	3.3	1800	RL
AmoebaNet-B [26]	CIFAR10	2.55 ± 0.05	-	2.8	3150	evolution
PNAS [18][1]	CIFAR10	3.41 ± 0.09	-	3.2	225	SMBO
ENAS [25]	CIFAR10	2.89	-	4.6	0.5	RL
NAONet [21]	CIFAR10	3.18[1]	15.67	10.6	200	NAO
DARTS [20]	CIFAR10	3.00	17.76	3.3	1.5	gradient
SNAS (moderate) [34]	CIFAR10	2.85	-	2.8	1.5	gradient
ProxylessNAS [3][2]	CIFAR10	2.08	-	5.7	4	gradient
P-DARTS [4]	CIFAR10	2.50	16.55	3.4	0.3	gradient
P-DARTS [4]	CIFAR100	2.62	15.92	3.6	0.3	gradient
ASAP [24]	CIFAR10	2.49 ± 0.04	15.6	2.5	0.2	gradient
PC-DARTS [37]	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)[3]	-	**1.68**	**13.03**	7.2	-	gradient

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

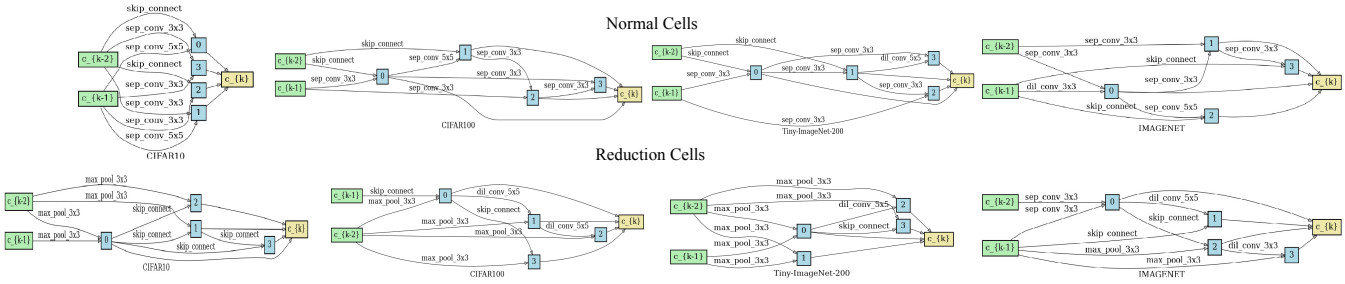

Figure 4: The cell of best structures searched on different datasets.

Architecture	Test Err. (%)	Params (M)
ResNet18 [38]	47.3	11.7
DenseNet-BC [17]	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

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

ment [5] and mixup [39] to achieve better results. Table 1 shows that DARTS+ achieves impressive 1.68% test error on CIFAR10 and 13.03% test error on CIFAR100, demonstrating the effectiveness of DARTS+.

Results on Tiny-ImageNet-200. The network is similar as 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 transfer the architectures searched from other algorithms to Tiny-ImageNet-200 and evaluate the performance for fair comparison. Other experimental settings are the same as CIFAR10/100.

The results are shown in Table 2. DARTS+ achieves the state-of-the-art 28.3% test error, which is much better than other baselines. Note that architecture searched on Tiny-ImageNet-200 with DARTS+ performs much better than DARTS and its parameter size is much larger than DARTS, because DARTS suffers from collapse and the architecture searched with DARTS contains lots of *skip-connect*s. Desired architectures are more likely to be generated with the "early stopping" paradigm.

Results on ImageNet. We use the architecture searched directly from ImageNet for evaluation, and the architecture from CIFAR100 to test the tranferability 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

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×) [28]	25.3	-	6.9	585	-	manual
ShuffleNet-V2 (2×) [22]	25.1	-	7.4	591	-	manual
NASNet-A [40]	26.0	8.4	5.3	564	1800	RL
AmoebaNet-C [26]	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 [20]	26.7	8.7	4.7	574	4.0	gradient
SNAS (mild) [34]	27.3	9.2	4.3	522	1.5	gradient
ProxylessNAS [3]†	24.9	7.5	7.1	465	8.3	gradient
P-DARTS (CIFAR10) [4]	24.4	7.4	4.9	557	0.3	gradient
ASAP [24]	26.7	-	-	-	0.2	gradient
XNAS [23]	24.0	-	5.2	600	0.3	gradient
PC-DARTS [37]†	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

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.).

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), 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.

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 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 a good architecture with 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 [39] for training to obtain 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.

4.4 Effectiveness of Early Stopping

To further verify the effectiveness of DARTS+, we conduct extensive experiments on selected architectures at different epochs. The classification results on CIFAR10, CIFAR100 and Tiny-ImageNet-200 are shown in Fig. 2. We also point out the time to "early stop" under two criteria, marked 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" is able to generate good architectures at both stopping criteria, regardless of the type of datasets.

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

5 Conclusion

In this paper, we conduct comprehensive analysis and extensive experiments to show that DARTS suffers from the collapse problem, which is mainly caused by the cooperation and competition problem in the bi-level optimization in DARTS. We propose the "DARTS+" algorithm, 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 the state-of-the-art performances on all benchmark datasets. Moreover, it should be noticed that many recent progresses of DARTS could use "early stopping" to achieve better results, and the proposed "early stopping" criteria could be applied to many other types of search spaces, including the RandWire search space [35], mobile convnets search space [30], etc.

References

[1] Arjovsky, M., and Bottou, L. 2017. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*.

[2] Brock, A.; Lim, T.; Ritchie, J. M.; and Weston, N. 2017. Smash: one-shot model architecture search through hypernetworks. *arXiv preprint arXiv:1708.05344*.

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

[4] Chen, X.; Xie, L.; Wu, J.; and Tian, Q. 2019. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. *arXiv preprint arXiv:1904.12760*.

[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] Ghiasi, G.; Lin, T.-Y.; and Le, Q. V. 2019. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *CVPR*, 7036–7045.

[8] Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *NIPS*.

[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] Joglekar, M. R.; Li, C.; Adams, J. K.; Khaitan, P.; and Le, Q. V. 2019. Neural input search for large scale recommendation models. *arXiv preprint arXiv:1907.04471*.

[15] Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

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

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

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

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

[22] 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.

[23] Nayman, N.; Noy, A.; Ridnik, T.; Friedman, I.; Jin, R.; and Zelnik-Manor, L. 2019. Xnas: Neural architecture search with expert advice. *arXiv preprint arXiv:1906.08031*.

[24] Noy, A.; Nayman, N.; Ridnik, T.; Zamir, N.; Doveh, S.; Friedman, I.; Giryes, R.; and Zelnik-Manor, L. 2019. Asap: Architecture search, anneal and prune. *arXiv preprint arXiv:1904.04123*.

[25] Pham, H.; Guan, M. Y.; Zoph, B.; Le, Q. V.; and Dean, J. 2018. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*.

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

[27] 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).

[28] 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.

[29] Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15(1).

[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. *arXiv preprint arXiv:1904.02877*.

[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. *arXiv preprint arXiv:1905.11946*.

[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. 2018. Snas: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926*.

[35] Xie, S.; Kirillov, A.; Girshick, R.; and He, K. 2019. Exploring randomly wired neural networks for image recognition. *arXiv preprint arXiv:1904.01569*.

[36] 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*.

[37] 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*.

[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] Zhang, H.; Cisse, M.; Dauphin, Y. N.; and Lopez-Paz, D. 2017. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*.

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

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