Enhancing Differentiable Architecture Search: A Study on Small Number of Cell Blocks in the Search Stage, and Important Branches-based Cells Selection

Bedionita Soro
KAIST AI
South Korea
e-mail:sorobedio@kaist.ac.kr

Chong Song
KAIST AI
South Korea
songchong@kaist.ac.kr

Abstract

In recent years, differentiable neural architecture search (DARTS) method has attracted a lot of attention. This method has been proposed to reduce the search cost incurred when using reinforcement learning and evolutionary search strategies. Although several studies have been carried out to improve its performance, most of these existing methods share some common limitations: They use a stack of five to eight cells during the search process to find only two distinct cells. The usage of several cells significantly increases the computation cost of the search process. In this paper, to reduce the search time, we propose to decouple the structure of the architecture used during the search of optimal pair of cells from the final architecture by using only one normal and one reduction cells search architecture during the search stage and the same architecture structure as DARTS during the evaluation stage. We also address the stability and performance drop trade-off by inserting additional residual connection in parallel with every normal cell block. Additionally, adding A convolution skip connection to the evaluation architecture has been shown to improve the performance. Finally, we investigated the effect of searching optimal cell's operation from highly performing branches in the internal structure of every cell. Extensive experiments showed that the proposed method significantly reduces the search cost while achieving promising results on ImageNet, CIFAR-10, and CIFAR-100 compared to existing state-of-the-art methods on DARTS search space.

1. Introduction

NAS is a sub-field of AutoML[25, 1] that focuses on automatic design of efficient neural network architectures [23, 13]. NAS does not only reduce the burden of building efficient neural network architectures but also produces architectures that outperform handcrafted state-of-the-art methods [15, 21]. Early NAS was dominated by evolutionary search-based approaches [24, 12, 22, 17] followed by reinforcement learning(RL) based NAS(RL-NAS) [34, 2] which has gained a lot of attention. However, those methods require hundreds or thousands of GPUs days to explore their search spaces. Several approaches have been proposed to reduce the search cost among them weights sharing approaches[19] as well as transferable cells-based search approaches such as NASNet search space[35] have been widely adopted. Although those methods have reduced the search constraints, for small-scale research centers it remains challenging to obtain an optimal architecture in a GPU day. Recently, differentiable NAS approaches that exploit path optimization within multi-paths cell stack architectures in a continuous way led to significant reduction of the computation time. Differentiable neural architecture search (DARTS) [18] method and its subsequent variants exploit a fixed network topology structure with two type of cell operation: reduction cell nodes that reduces the spatial dimension of the input and normal cells nodes that does not changes the resolution of the input. In DARTS the reduction nodes are always located at 1/3 and 2/3 in the cell stack. Also, during the search stage, several cells are stacked to design the cell searching model which is trained to find only one normal and one reduction cell. Inevitably, the usage of several cells during the search stage significantly increases the search cost. Several methods have been proposed to improve the performance but most of them focus only on the choice of optimal operation for the two cells without considering the search network structure itself. Another interesting fact about DARTS approach and its variants is that random search can in some cases achieve better performance than the searched cells based model. Also, it has been demonstrated that the validation performance during the search stage is not an indication of the performance of the final model.

Few existing DARTS variant approaches such as [6] investigated those limitations. However, the number of identical cells in the cell search architecture were limited to 3 normal and 2 reduction cells. It would be interesting to investigate the effect of using only 2-cells based search architecture in the search stage. Unfortunately, in literature such a study has not yet been conducted.

In this paper, we propose to investigate the effect of using 2 cells-based architecture in the search stage by decouple the cell search network structure from the evaluation network structure. The 2 cell-based architecture consists of one normal cell and one reduction cell. With such a search network structure, we can reduce the search cost by 50%. However, using only 2 cells architecture in the search stage of the original DARTS produces searched cells with a predominance number of pooling operations leading to performance degradation. We reduce that problem by either inserting a convolution skip-connection block in parallel with the normal cell or restricting the number of pooling operations per cell. The convolution skip-connection can also be inserted in parallel with normal cells during the evaluation stage to improve the performance. The auxiliary skip-connect added is like the residual blocks in Resnet architecture[15].

Contribution: Our contribution can be summarized as follows:

- We propose an effortless way to reduce the search cost of differentiable NAS approaches based on DARTS search space.
- We show that augmenting the cells search architecture as well the evaluation architecture with an additive auxiliary convolution skip-connection leads to performance improvement.
- Additionally, we investigate the effect of searching best operations from top two performing branches at every node in the internal structure of the cells using evolutionary search.
- We prove through experiment that the proposed method significantly reduces the search cost while maintaining superior performance compared to existing DARTS approaches.

2. Related work

Differentiable NAS enables searching through gradient based optimization on both network's weights and architecture paths where the final architecture is defined based on the layer's operations with larger weights. This technique has two major problems [3]: the first is that the model often requires to compute the hessian of the loss to optimize the architecture's weights and that leads to longer search time and higher computation cost. The second problem is related to the choice of the important connections or paths because the important connections based on higher weights produces cell's structure with several residual or pooling operation that degrades the performance. Thereby, [29] introduced SNAS, a differentiable NAS method exploiting stochasticity in the search process, and they learned the architecture distribution that allowed to efficiently select the best operations. Similarly, [6] proposed to gradually increase the number of normal cells without changing the architecture structure. This produced diverse levels of architecture with better performance. [4] enabled the search with a DARTS-like architecture on large scale image dataset by proposing a method called proxylessnas that optimizes one architecture path at a time with the target device latency constraints included in the loss function. [26] explored and combined the benefit of non-differentiable and differentiable losses used in RL-NAS and DARTS, to propose a unified method called UNAS a method that takes advantage of differentiable NAS and RL-NAS for better performance. While several studies[33, 27, 9] focused on improving the performance of DARTS, some methods such as the one proposed by [30] attempts to reduce the burden introduced by the second-order optimization through the usage of partial channel connections where only a random subset of channels are sent to the mixed-operation. However, [5] showed that bypassing the channels in such a way leads to unstable selection of the operations. In the same direction, [28] proposed to use zeroth-order optimization to speed up the search training process. Although, all those methods achieve promising performance on DARTS search space, they all use fixed topology structure, and train multiple stacks of normal and reduction cells during the search stage. Therefore, the search for only one pair of normal and reduction cells requires a higher computation and memory resources than required by a network with only one pair of cells.

Cell-level Search Space. The search space defines the set of network operations (convolution, pooling, and linear layers) which are used to build the network architectures. The search space can be made of every individual operation or motif which consists of sets of operations organized as sub module or subnets[35] such as the building blocks in ResNet and InceptionNet. Cell-level NAS search space has achieved promising results with less search cost and have been largely used in recent NAS algorithms[11, 8]. Similarly, we focus our study on DARTS search space which consists of searching the structure of two cells. Each cell consists of 4 nodes where each edge is a weighted combination of all operations in the search architecture and the evaluation network's cells consist of nodes with two operations per node. The internal structure of cell in DARTS search space is presented in Figure 1.

3. Method

3.1. DARTS Description

In this subsection, we present an overview of the original DARTS approach to which our method is closely related. DARTS instead of searching over a discrete search space, relaxes the search space to be continuous so that the cell's topology can be optimized through gradient decent with respect to the training loss. In Figure 2 we present the main architecture of DARTS. There are two types of cells in DARTS architecture: normal and reduction cells. A normal cell consists of a set of operations with two inputs that it transforms through several operations and then combines their results to produce one output feature map without changing the spatial resolution of the input feature map. The reduction cell on the other hand processes the data similarly but reduces the input spatial resolution by a factor of 2 using stride 2. The skip-connection like link in the architecture is a skip-input representing the second input of the current cell and not an additive residual connection. The search space of DARTS consists of seven operations: *Max-Pooling*, *skip-Connection*, *Average-Pooling*, *Separable-3x3-convolution*. *Separable-5x5-convolution*, Dilated-3x3-convolution, and *Dilated-5x5-convolution*. *None* is added as the eighth operation.

The internal structure of a cell in DARTS architecture can be viewed as a directed acyclic graph with four nodes as presented in Figure 1 where every node receives the feature maps of its predecessors. Each edge in the internal structure of the cell consists of a weighted combination of all the seven operations in the search space. The nodes are often called feature maps. We denote the i$^{\text{th}}$ and j$^{\text{th}}$ node or feature map by $x^{(i)}$ and $x^{(j)}$ and the edge operation between them by $o^{(i,j)}$ which represents the network operations that transform the feature map $x^{(i)}$ into $x^{(j)}$. The weights or coefficients of the edge's operation are represented by $\alpha_N \in \mathcal{R}^{14\times 8}$ for the normal cell and $\alpha_R \in \mathcal{R}^{14\times 8}$ for the reduction cell. We will refer to those weights as α, hyper-parameters a, mixing coefficients or architecture weights. The feature map at each intermediate node is produced using the weighted combination of its predecessors' feature map as shown in eq.1

$$x^{(j)} = \sum_{o\in O,\quad i<j} a^{(i,j)} o^{(i,j)}(x^{(i)}). \tag{1}$$

The mixing coefficients or weights are defined as follows:

$$a^{(i,j)} = \frac{exp(a_o^{(i,j)})}{\sum_{o\in O} exp(a_o^{(i,j)})}, \tag{2}$$

where a_o denotes the coefficient associated with the operation o.

Since each internal link of cells is weighted combination of all operations, the first step in DARTS consists of training a network with eight cells while optimizing the network weights and the operations' mixing coefficients. This training is carried out through a bi-level optimization where the main goal is to drop the multi-paths structure of the internal structure of cells by retaining only the best two edge operations between nodes as shown in Figure 1. The bi-level optimization problem is defined as in eq.3.

$$\begin{aligned} \min_{a} \quad & L_{val}(w^*(a), a) \\ \text{s.t.} \quad & w^*(a) = argmin_w L_{train}(w, a) \end{aligned}, \tag{3}$$

where L_{val} is the validation loss and L_{train} the training loss. the networks weights are represented by w.

After solving this optimization problem, for each cell, the operations corresponding to the top-2 larger mixing coefficients(architecture weight) are chosen to construct the final cell Figure 1. The resulting cells are used to build the final model which is trained from scratch on the target dataset.

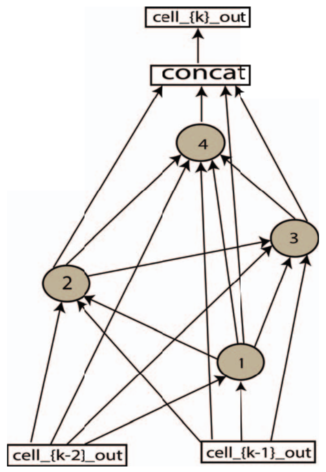

Figure 1. Cell internal structure. The circle with numbers represent the nodes $x^{(i)}$. except the links between the concat box and other nodes each link represents a set of all operations which outputs are sum up at each node as defined eq.1

3.2. Decoupling cell search from architecture topology search

In this section, we described the major modifications we introduced in DARTS model.

The current section involves investigating the usage of two-cells structure network during the search for optimal pair of cells in DARTS. Figure2 shows the proposed search architectures (Figure 2-b& c) compared to the commonly used search architecture in DARTS and its variants(Figure 2-a).

We focus on reducing the search cost through modifying the cell search architecture while keeping the optimization

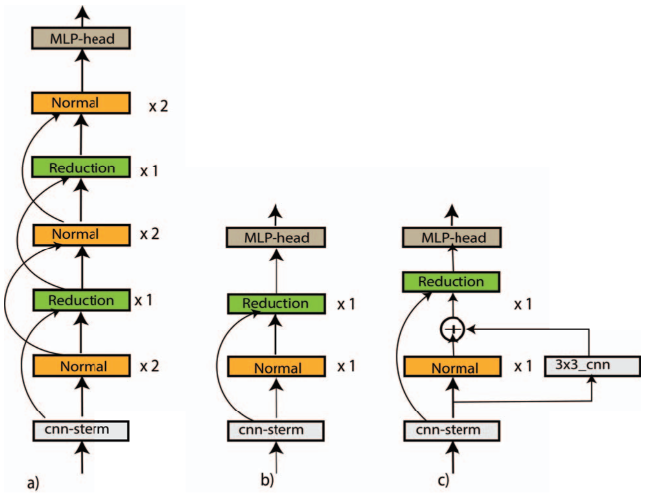

Figure 2. DARTS search topology structure: **a):** DARTS [18], PC-DARTS [30] and others search network architecture. **b)** and **c)**: Our search architectures. The convolution sterm (a 3x3-convolution layer) and the multi layer percetron(MLP head) are the same for all. $\times 2$: means that this cell is repeated 2 times.

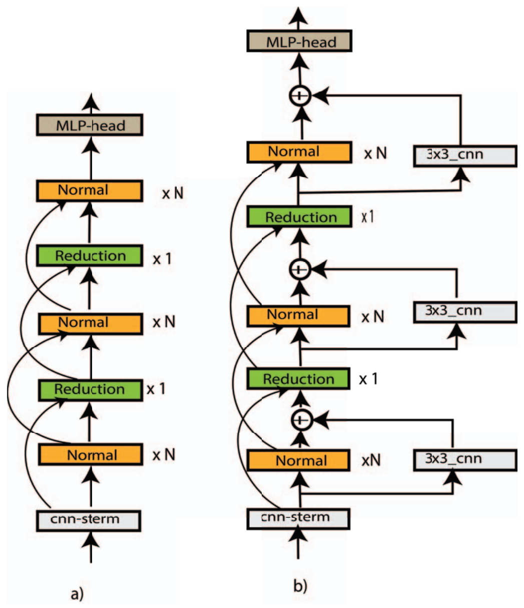

Figure 3. DARTS search topology structure: **a):** DARTS [18], PC-DARTS [30] and others search network architecture. And **b):** Our search architectures. The convolution sterm (a 3x3-convolution layer) and the multi layer percetron(MLP) are the same for all. $\times N$: means that this cell is repeated N times.

process the same as previous works. The search architecture is mainly composed of one normal cell and one reduction cell. Extensive experiments are conducted to identify the trade-off of using two-cell stack network in the search stage through which we found that our approach when applied with second order DARTS tends to produce cells with a predominance of pooling operations resulting in poor performing searched architecture. This predominance of pooling operations may be due to the small structure of the search network. To reduce this predominance of pooling operations, we propose to restrict the number of pooling operations in a cell to no more than 2. This restriction is Done during the operation selection as follows:

1. set pooling counter to zero
2. select the node and choose the top-2 operations and assign the corresponding operation to node 0.
3. Increase the counter by the number of pooling operations selected in previous operation
4. Repeat step 2 with the next node. if the maximum number of pooling operation is reached and there is a pooling operation in corresponding top-2 weights then select the next best top-2 non-pooling operations. Otherwise, select the top-2 weights corresponding operations and increase the pooling counter according to step 3.
5. Repeat previous step until two operations are assigned to each node.

The top-2 weights have the higher weights in the list of mixing (architecture weights) coefficients defined in eq.2.

Additionally, we found through experiments that inserting an auxiliary convolution residual connection in parallel with cells reduces the predominance of pooling operations in the cells and leads to high performing architecture. Similar to Resnet, the introduction of auxiliary residual connection does not increase the training cost during the search stage.

Furthermore, we similarly modify the standard searched, or evaluation network topology by augmenting the architecture with auxiliary convolution residual connection as shown in Figure 3. Due to the higher number of normal cells in the searched architecture, inserting an auxiliary connection for every normal cell leads to higher number of parameters and FLOPS. Nonetheless, the augmented evaluation architecture improves the performance of the based architecture. The proposed modifications can be applied to most existing DARTS variant methods to improve the performance or to speed up the search while reducing the computation cost during the search stage.

3.3. Optimal branches based search

One motivation behind the usage of important architecture weights a based selection is to select the top-2 operations that maximize the validation accuracy by pruning the remaining operations. In such way operations with smallest hyper-parameter a(architecture weights) are assumed to have negligible contribution. However, important hyper-parameters a based operations selection does not explore the case whether the selected operations are from the top performing branches at each node. As shown in Figure 4, where the top-2 higher hyper-parameters are indicated by red and blue arrow in bold which directly determine the operations and the branches to choose although the same operations selected from others branches could lead to better performance.

In this section, we explore the contribution of each branch in the internal structure of every cell to identified the top performing branches before selecting the top-two operations. The contribution of each cell's internal branches is identified using evolutionary search. The procedure consists of introducing internal branches selection weights which are fixed to 1 during the cells search stage. At the end of the cell search network training loop, the top two edges or branches with the higher architecture weights a have their branch weights set to 1.0 and the remaining branches weights are set to 0 for every node in each cell. The best branches search is done using evolutionary search as in algorithm 1. The branches weights are binary coded and only two branches are activated at a time for each node as shown in eq.4

$$\begin{aligned}\beta = [&[1.0, 1.0],\\ &[0.0, 1.0, 1.0],\\ &[1.0, 0.0, 1.0, 0.0],\\ &[1.0, 0.0, 1.0, 0.0, 0.0]]\end{aligned} \quad , \tag{4}$$

where each row decodes the two branches to be selected by 1 and 0 the non selected branches in the internal structure of each cell Figure1. The branches weights transform eq.1 to equation 5.

$$\bar{x}^{(j)} = \sum_{o\in O, \quad i<j} \beta(j)a(i,j)o^{(i,j)}x^{(i)}. \tag{5}$$

In the evolutionary search, single point crossover and list rotation based mutation operations are used. The evolutionary search is only applied to the branches weights β for each cells. The model used during the search stage is the one described in Figure 2-b. In the evolutionary search algorithm 1, a population is an architecture with two cells in total encoded as in eq.4. The evolutionary search is applied directly after the end of the search architecture optimization loop with initial population set to 20, and is run for 100 generations. The fitness function is the evaluation performance where only two branches are selected at every node in the internal structure of each cell.

Algorithm 1 Searching from best branch

Input: training data
Parameter: Optional list of parameters
Output: top best cells

```
1: Create the search network
2: Set nodes branches weights $\beta$ to 1($\beta = 1$)
3: while not converged do
4:     $\alpha \leftarrow \nabla_w \mathcal{L}_{val}(\omega - \xi \nabla_w \mathcal{L}_{train}(\omega, \alpha), \alpha)$
5:     $\omega \leftarrow \omega - \eta \nabla_w \mathcal{L}_{train}(\omega, \alpha)$
6: end while
7: Freeze architecture and model's weights($\alpha$ and $\omega$)
8: Initialize the population size $n$
9: set Elitist size($e$)
10: set number of generation($m$)
11: Initialize $\beta$ based on final weights $\alpha$
12: Create $n$ architectures ($\mathcal{P}$ using crossover and mutation
    operations on $\beta$ for each cell.
13: Evaluate all the architectures in $\mathcal{P}$.
14: $i \leftarrow 0$
15: while $i < m$ do
16:     Sort descent $\mathcal{P}$ and remove $top_e$ architectures de-
        fined as $\mathcal{E}$
17:     Generate $n - e$ new set ($\mathcal{Q}$) of architectures with
        crossover and mutations
18:     Evaluate $\mathcal{Q}$
19:     $\mathcal{P} \leftarrow \mathcal{Q} \cup \mathcal{E}$
20:     $i \leftarrow i + 1$
21: end while
22: Return $top_e$ best architectures
```

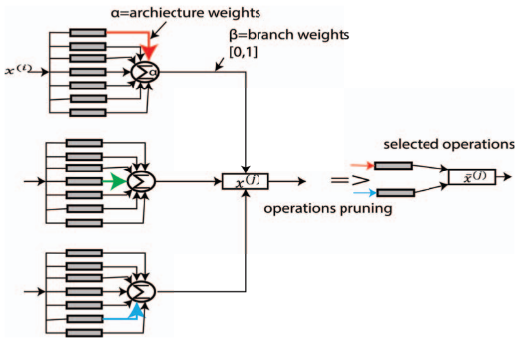

Figure 4. Process to select the operations in DARTS where the top two important architecture weights a are repsented by the red and the blue arrows. α is the architecture weights or coefficients defined in eq.2, β is the branch weights defined in eq.4. $x^{(j)}$ represents operation performed by eq.5 and $\bar{x}^{(j)}$ denotes the final state of the node.

4. Experiments

Datasets: The search and evaluation is performed on both small and large scale datasets such as CIFAR-10, CIFAR-100, [16], and ImageNet-1k [20] datasets. CIFAR-

10 and CIFAR-100 datasets consist of 60,000 with spatial resolution 32x32 colour images with respectively 10 and 100 classes. In both cases there are 50,000 training images and 10,000 testing images. The spatial resolution of the images are 32×32. ImageNet-1k [20] is the most highly-used subset of ImageNet large scale image classification and localization dataset. It has 1000 classes and contains 1,281,167 training images, and 50,000 validation images.

Training setup: Stochastic gradient descent optimizer with 3e-4 weights decay is used to optimize the model weights with initial learning rate 0.025. Cosine learning rate scheduler is used with momentum of 0.9. The architecture hyperparameters a optimization learning rate is set to 3e-4 with weights decay set to 1e-3.

During the search stage, we follow the standard search and evaluation as in [18, 30]. However, for the search stage we use architectures that consists of 2 cells(one normal cell and one reduction cell) which is trained for 50 epochs. The searched architecture consists of a stack of 20 cell(18 normal 2 reduction) for CIFAR10 and CIFAR100 dataset. While for ImageNet we used 14 cells(12 normal and 2 reduction). Our modification are applied to DARTS [18] and PC-DARTS[30] approaches. In addition to the commonly used evaluation architecture of DARTS we also evaluate the performance of the augmented architecture.

Baselines: We assess the quality of proposed search strategies through extensive experiments in which the proposed method is applied to DARTS, and PC-DARTS. DARTS, PC-DARTS, FairDARTS [7], and some recent high performing DARTS variants are used as baselines.

Metrics: The evaluation metrics are the search duration in days denoted by g-cost or gpu-cost or gpu-days, the performance metric is the classification accuracy. # cell denotes the number of cells in the search networks.

4.1. Evaluation on CIFAR-10 and CIFAR-100 datasets

We evaluate the quality of the search strategy and searched cells on CIFAR-10 and CIFAR-100.

setup: During the search stage on CIFAR10 and CIFAR-100, 50% of the training data is used as validation sets. The evaluation model is trained on the full training dataset for 600 epochs on one Nvidia GPU Titan RTX-100 for each dataset. After completing the search stage, we choose the last best pair of cells to construct the evaluation architecture which is then trained from scratch.

Results: The results presented in both Tables 1 and 2 show the proposed method significantly reduces the search time up to between 50% and 80%. However, We observe that the proposed search method with no auxiliary residual connection is less stable compared to previous variants of DARTS. Which means for some search runs the cells found may not outperform the baselines. Nonetheless, using the CNN auxiliary residual connection or restricting the number of pooling operations per cell during the search stage leads to much better performance. We also observe that the search for CIFAR-10 on CIFAR-100 dataset produces highly performing architecture for CIFAR-10 than searching directly on CIFAR-10 dataset.

Models	top1	g-cost	search arch.	params.	search on
Random	96.75±0.18	–	8 cells	3.4	CIFAR-10
DARTS_v2[18]	97.24±0.09	3-4	8 cells	3.5	CIFAR-10
PC-DARTS[30]	97.43 ± 0.06	0.4	8 cells	3.6	CIFAR-10
P-DARTS[6]	97.50	0.3	5 cells	3.4	CIFAR-10
FairDARTS[7]	97.41± 0.14	0.4	8 cells	3.8	CIFAR-10
β-DARTS[31]	97.47±0.08	0.4	8 cells	3.8	CIFAR-10
CyDAS[32]	**97.52±0.04**	0.3	8 cells	3.9	CIFAR-10
DARTS_V2^{++}(ours)	97.02±0.03	0.3	2 cells	3.4	CIFAR-10
DARTS_V2^{++}(ours restrict)	97.27±0.05	0.3	2 cells	3.5	CIFAR-10
DARTS_V2^{++}(ours)	97.31±0.05	0.29	2 cell+skip	3.5	CIFAR-10
DARTS_V2^{++}_aug(ours)	97.44±0.08	0.3	2 cells	8.8	CIFAR-10
PC-DARTS^{++}(ours)	97.07±0.07	**0.08**	2 cells	3.9	CIFAR-10
DARTS_V2^{++}_aug(ours)	97.45±0.05	0.29	2 cell+skip	9.1	CIFAR-10
DARTS_V2^{++}(ours restrict)	97.51±0.07	0.3	2 cells	9.6	CIFAR-10
DARTS_V2^{++}(ours)	97.52±0.03	0.3	2 cells	3.7	CIFAR-100
DARTS_V2^{++}(ours)	**97.60±0.05**	0.3	2 cells	9.6	CIFAR-100

Table 1. Classification results on CIFAR-10 dataset with predefined topology. Model_aug means the evaluation model used skip connection. The search cost(g-cost) is in GPU-days. DARTS_V2 restrict is the case where we restrict the number of pooling operation to no more than 2.

Models	top1	g-cost	#cells	params(M)	search on
DARTS_v2[18]	82.46	3.4	8	3.4	CIFAR-10
PC-DARTS[30]	83.1	0.1	8	3.6	CIFAR-100
P-DARTS[6]	83.45	0.3	5	3.6	CIFAR-10
GDAS[10]	81.62	4	8	3.4	CIFAR-10
β-DARTS[31]	83.48±0.03	0.4	8	3.8	CIFAR-100
CyDAS[32]	**84.31**	0.3	8	3.9	CIFAR-10
DOTS[14]	83.52±0.13	0.26	8	4.1	CIFAR-100
DARTS_V2^{++}	82.17±0.17	0.3	2	4.5	CIFAR-100
DARTS_V2^{++}_aug	83.58±0.22	0.29	2	10.4	CIFAR-100
PC-DARTS^{++}	83.01±0.35	**0.08**	2	≈ 4.0	CIFAR-100

Table 2. Classification results on CIFAR-100 dataset with predefined topology. Model^{++} means the search used only one pair of cells. The search cost(G-COST) is in GPU-days.

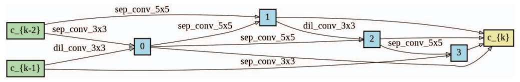

Figure 5. Normal cell found on CIFAR-10

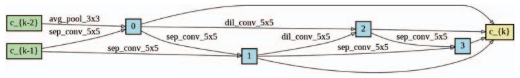

Figure 6. Reduction cell found on CIFAR-10

4.2. Evaluation on ImageNet

In this task, we assess the quality of our search strategy on ImageNet dataset.

Setup: We used similar setting as in section 4.1 for the search and evaluation. The searched model is trained on 4 Nvidia Titan RTX-100 GPUs for 250 epochs while the search has been done on 1 to 2 Nvidia Titan RTX -100.

Results: Table 3 presents the results of this experiment. We observed that the proposed search approach applied with PC-DARTS on ImageNet discovered architectures that outperform the original PC-DARTS with less search time and same number of training epochs and settings . With two-cell based search network the search process speed up to 64% faster on ImageNet with a maximum top-1 performance up to 75.2 and 76.3% compared to DARTS and PC-DARTS which only achieved respectively top-1 accuracy of 73.3% and 74.9%. This experiments results have demonstrated that using one pair of cells stack network during the search stage is much preferable than the original PC-DARTS, and DARTS architectures on ImageNet.

Models	top1	top5	cost	#cells	Searched on	Params(M)
DARTS_v2[18]	73.3	91.3	4	8	CIFAR-10	4.7
PC-DARTS[30]	74.9	92.2	3.9	8	ImageNet	5.3
P-DARTS[6]	74.9	92.3	0.3	5	CIFAR-10	5.1
FairDARTS-C[7]	**77.2**	**93.5**	3	8	ImageNet	5.3
DOTS[14]	76.0	92.8	1.3	8	ImageNet	5.3
CyDAS[32]	76.3	92.9	1.7	8	ImageNet	6.1
DARTS^{++}(ours)	74.3	91.7	**0.3**	2	CIFAR-10	5.3
DARTS^{++}_aug(ours)	75.2	92.2	0.3	2	CIFAR-10	12.3
PC-DARTS^{++}_aug(ours)	76.1	92.8	1.4	2	ImageNet	7.24
PC-DARTS^{++}(ours)	76.3	92.7	**1.4**	2	ImageNet	5.58

Table 3. Classification results on ImageNet dataset with predefined topology. Model^{++} means the search used only one pair of cells. The search cost(G-COST) is in GPU-days. The search is run once in this experiment

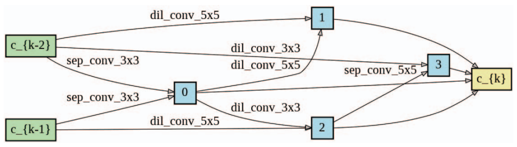

Figure 7. Normal cell found on Imagenet

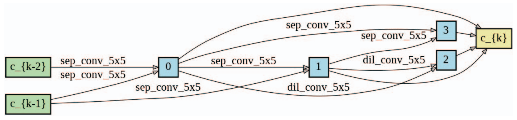

Figure 8. Reduction cell found on Imagenet

4.3. Optimal branches based search

Setup: In this section, we investigate whether selecting the top best branches based on the higher hyper-paramters a or architecture weights a is the optimal choice of operation in DARTS. After the hyper-parameters a optimization, evolutionary search is used to search for the top-2 branch that maintain a good validation accuracy for each cell then the important hyper-parameters a process is used to select the best operations from those branches. In this experiment the best branches search is done on two-cells stack network.

Results: Table 4 presents the results of the experiment with and without evolutionary search. the column #cells denotes the number of cells in the search stage network. The first two rows present the results of the final architecture based on the cells found using the higher hyper-parameters a to select the internal operations of each cell. The bottom two row denote the accuracy of the same search architecture but using to the evolutionary search to find the top two best branch for every cell. In Table 4, we observe that after the optimization of the search architecture applying evolutionary search to find the best performing branches then select the top-2 operations based on important parameters a leads to better performance improvement when using PC-DARTS baseline but that has little effect with DARTS structure based model. Although searching for optimal branches evolutionary search after the search architecture optimization does not significantly improve the performance when using DARTS baseline, it enables finding much more stable architectures with comparable performance. The search for branches contribution can be used to search for heterogeneous cells. In fact, to search for architecture with several type of normal and reduction cells, after the optimization of the search architecture one can stack the two cells found to build the final architecture then apply evolutionary search to selects the top 2 best branches for each cell independently. This will require conducting evolutionary search with a large number of cells which will introduce more additional search time than for two cell based structure. Nonetheless, this results of this approach may open the door for more efficient two-cells based search and high performing heterogeneous DARTS architectures.

Models	top1	Total search cost	#cells	params(M)	search on
DARTS_v2^{++}	97.14	0.3	8	3.8	CIFAR-10
PC-DARTS^{++}	96.87	0.08	2	4.2	CIFAR-10
DARTS_V2^{++}(w)	97.15	0.57	2	4.0	CIFAR-10
PC-DARTS^{++}(w)	97.16	0.16	2	4.1	CIFAR-10

Table 4. Ablation study results; The first two rows contain the results of the important hyper-parameters a based operations or branches selection, and the bottom two rows show the results of the final architure build based on evolutionary search based operations selection found from the same optimized search network.

5. Conclusion

In this paper, we investigated the effect of using only one normal and one reduction cell in the search stage architecture of DARTS and its variant PC-DARTS. We augmented the search architecture with an auxiliary convolution skip connection that allows finding high performing cells. We also proposed an augmented version of the evaluation architecture. Through extensive experiments, the proposed method significantly reduced the search cost while achieving comparable performance to existing methods. Additionally, we investigated the search while considering the contribution of each branch in the internal structure of each cell. Such an method has shown to improve the performance on PC-DARTS based structure. This study may open the door for more efficient search methods for differentiable NAS.

Acknowledgements

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2019-0-00075, Artificial Intelligence Graduate School Program(KAIST))

References

[1] Hutter F. Caruana R. Bardenet R. Bilenko M. Guyon I. Kegl B. and Larochelle H. Automl 2014 workshop. In *Proceedings of the International Conference on Machine Learning (ICML 2014)*, 2014.

[2] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. *ArXiv*, abs/1611.02167, 2017.

[3] Kaifeng Bi, Changping Hu, Lingxi Xie, Xin Chen, Longhui Wei, and Qi Tian. Stabilizing darts with amended gradient estimation on architectural parameters. *ArXiv*, abs/1910.11831, 2019.

[4] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *ArXiv*, abs/1812.00332, 2019.

[5] Xiangning Chen, Ruochen Wang, Minhao Cheng, Xiaocheng Tang, and Cho-Jui Hsieh. Dr{nas}: Dirichlet neural architecture search. In *International Conference on Learning Representations*, 2021.

[6] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1294–1303, 2019.

[7] Xiangxiang Chu, Tianbao Zhou, Bo Zhang, and Jixiang Li. Fair DARTS: Eliminating Unfair Advantages in Differentiable Architecture Search. In *16th Europoean Conference On Computer Vision*, 2020.

[8] Xuanyi Dong, Lu Liu, Katarzyna Musial, and Bogdan Gabrys. NATS-Bench: Benchmarking nas algorithms for architecture topology and size. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2021.

[9] Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1761–1770, 2019.

[10] Xuanyi Dong and Yezhou Yang. Searching for a robust neural architecture in four gpu hours. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1761–1770, 2019.

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

[12] Jan Drchal and Miroslav Snorek. Genetic programming of augmenting topologies for hypercube-based indirect encoding of artificial neural networks. In *SOCO*, 2012.

[13] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *J. Mach. Learn. Res.*, 20:55:1–55:21, 2019.

[14] Yuchao Gu, Yun Liu, Yi Yang, Yu-Huan Wu, Shao-Ping Lu, and Ming-Ming Cheng. Dots: Decoupling operation and topology in differentiable architecture search. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12306–12315, 2021.

[15] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[16] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[17] Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical representations for efficient architecture search. *ArXiv*, abs/1711.00436, 2018.

[18] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *ArXiv*, abs/1806.09055, 2019.

[19] Hieu Pham, M. Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In *ICML*, 2018.

[20] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[21] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2015.

[22] Kenneth O. Stanley. A hypercube-based indirect encoding for evolving large-scale neural networks. 2009.

[23] Kenneth O. Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10:99–127, 2002.

[24] Kenneth O. Stanley and Risto Miikkulainen. Efficient evolution of neural networks through complexification. 2004.

[25] Chris J. Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Auto-weka: combined selection and hyperparameter optimization of classification algorithms. *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013.

[26] Arash Vahdat, Arun Mallya, Ming-Yu Liu, and Jan Kautz. Unas: Differentiable architecture search meets reinforcement learning. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11263–11272, 2020.

[27] Ruochen Wang, Minhao Cheng, Xiangning Chen, Xiaocheng Tang, and Cho-Jui Hsieh. Rethinking architecture selection in differentiable nas. In *International Conference on Learning Representations (ICLR)*, 2021.

[28] Xiaoxing Wang, Wenxuan Guo, Junchi Yan, Jianlin Su, and Xiaokang Yang. Zarts: On zero-order optimization for neural architecture search, 2022.

[29] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: Stochastic neural architecture search. *ArXiv*, abs/1812.09926, 2019.

[30] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. Pc-darts: Partial channel connections for memory-efficient architecture search. In *ICLR*, 2020.

[31] Peng Ye, Baopu Li, Yikang Li, Tao Chen, Jiayuan Fan, and Wanli Ouyang. b-darts: Beta-decay regularization for differentiable architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10874–10883, June 2022.

[32] Hongyuan Yu and Houwen Peng. Cyclic differentiable architecture search. *IEEE transactions on pattern analysis and machine intelligence*, PP, 2022.

[33] Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. Understanding and robustifying differentiable architecture search. *ArXiv*, abs/1909.09656, 2020.

[34] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. *ArXiv*, abs/1611.01578, 2017.

[35] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8697–8710, 2018.