DPC-DARTS: Architecture diversify for partial channel connections-differentiable architecture search

Yichao Liu, Farzad Nozarian,
Igor Vozniak, Christian Müller
Deutsches Forschungszentrum für
Künstliche Intelligenz (DFKI),
University of Saarland,
Saarland Informatics Campus D 3.2,
66123 Saarbrücken, Germany
yichao.liu@dfki.de

Yueyang Teng
Sino-Dutch Biomedical and
Information Engineering School,
Northeastern University,
110004 Shenyang, China
tengyy@bmie.neu.edu.cn

Abstract

The differentiable architecture search (DARTS) has greatly improved the efficiency of neural architecture search by applying gradient-based optimization. However, both the normal cells and reduction cells in the structure are sharing the same cell structure. This kind of single structure destroys the diversity of DARTS. Hence, this paper is proposed to address this issue by introducing channel-wise batch normalization. We propose a novel method which helps to extremely reduce number of parameters of the network and as a side effect improves the training stability with faster network convergence and lower memory consumption in comparison to the original DARTS. The conducted experimental based on CIFAR10 and CIFAR100 data sets reveal high performance compared to state of art methods.

Keywords: neural architecture search, differentiable architecture search, partial channel connection, batch normalization, diversity

1 Introduction

Discovering an efficient and high performance handcrafted neural network is a time consuming task. Even if the unprecedented CNN models are designed, the high performance hardware requirement for training on the large-scale datasets is hard to meet. Especially in the era of IoT, where the challenge is to both keep the high performance and to reduce the inference time. In order to tackle these challenges neural architecture search (NAS) come into play. Research works on NAS for the last couple of years shown a revolutionary results in improving both the search time and efficient use of GPU's memory. (Stamoulis et al., ; Cai et al., 2018; Xu et al., 2019; Liang et al., 2019; Liu et al., 2018).Results on NAS shows that automatically designed architectures can compete with the handcrafted one in various tasks of classification (Stamoulis et al., ; Cai et al., 2018; Chen et al., 2019; Fang et al., 2019), object detection (Howard et al., 2019; Tan et al., 2019).

Previous gradient-based research works were concentrated on one-shot methods (Cai et al., 2018; Liu et al., 2018; Xie et al., 2018), most of which require a lot of time to search and train large number of parameters, where the number of compute operations is determined by the number of parameters, which obviously defines the inference time and memory requirements(Liu et al., 2017). Despite the recent progress, even the DARTS method, one of the latest and advanced NAS, still requires a lot of time, namely days, in order to search for a compact and efficient network architecture.

In this paper, we propose a novel method based on DARTS to reduce both search time and number of parameters of searched network, thus it can be easily deployed in the practical application. The core idea comes from NAS (Xu et al., 2019) and network pruning (Liu et al., 2017). Unlike PC-DARTS and DARTS, we can avoid alternative updating the weights and architecture parameters. This makes it easy to implement without freezing any parameters during the entire searching progress. With the penalty of batch normalization scaling factors, the operation-wise sparsity is highly improved, which leads to fastest optimal operation selection.

In order to highlight performance improvement, we present some typical classification experiments. The paper is organized as follows: Section 2 covers related work overview. , where the proposed method is described in section 3. Section 4 highlights the experiments conducted on CIFAR10/100 (Krizhevsky et al., 2009) and the conclusion is given in section 5.

Our contributions can be summarized as follows:

1. A novel method for diversifying original DARTS structure, instead of stacking the same reduction cell and normal cell.

2. By introducing scaling factor, the alternative upgrading can be avoided. Thus, the search optimization can be done easier.

3. The proposed novel method, guarantees extreme reduction of network parameters. As such, integration with the embedded devices to achieve a very concrete goal, like image classification is more realistic and less expensive.

4. Compared with original DARTS and other NAS methods, it uses less time for the search and allows increase of batch size.

2 Related works

The manually designed neural networks, has unlocked unprecedented performance in solving computer vision problems. However, higher accuracy usually accompany with significant computational requirements (Krizhevsky et al., 2012; Simonyan and Zisserman, 2014; Szegedy et al., 2015; He et al., 2016), In order to tackle high computational demands problem methods-neural architecture search (NAS) has been researched. NAS is a common technique in order to automatically design simple neural networks based on reinforcement learning (RL), evolutionary algorithms, and one-shot algorithms (Liu et al., 2018; Tan et al., 2019; Pham et al., 2018; Real et al., 2019; Gong et al., 2019). Recent works shows that RL and evolutionary algorithms are computationally expensive to run, despite reduced time (Pham et al., 2018; Brock et al., 2017). The proposed in this paper work is mostly related to one-shot family of methods, which trains a network with gradient-based optimization that comprises all the candidate operations (Zoph et al., 2018). The ProxylessNAS (Cai et al., 2018) tries to get rid of the meta-controller and also reduce the memory consumption based on the thought of path binarization. The DenseNAS (Fang et al., 2019) on the other hand introduces a nice solution to the breath search problem by constructing a dense connection search space. However, adding block structures leads to the rapid increase in number of parameters. On the other hand, many research works are concentrated on reducing searching time of DARTS. In P-DARTS (Chen et al., 2019) author proposed to avoid bias to skip-connection operation during searching stage and bridge the depth gap between search and evaluation. MDENAS (Zheng et al., 2019) reduces the converge time by encoding the path/operation selection as a distribution sampling, which is a different approach compared to other one-shot methods. Highlighted methods are similar to DARTS and aim to reduce searching time, on the other hand their networks are still larger than the one proposed in DARTS. The main reason for significant larger number of parameters is the lack of layer wise diversity. In this work, we address mentioned problems through the batch normalization technique.

3 The Proposed Approach

3.1 DARTS Overview

The searched network is formed from reduction and normal cells, which are searched by DARTS structure. Each cell can be organized as a directed acyclic graph (DAG) of N nodes. The operational space can be denoted as O, in which many candidate operations are comprised such as, zero, skip-connect, convolution, average-pooling, etc. The edge between each pair of nodes represents the information propagated from x_i to y_i, which is weighted by hyper-parameter $d(i,j)$. In particular, a softmax function of architecture parameters for each edge, can be denoted as in Equation 1

$$p_{i,j} = \frac{exp(d^{o}_{i,j})}{\sum_{o' \in C} exp(d^{o'}_{i,j})} \tag{1}$$

which stands for a weight of each operation, where the entire formulation is given in Equation 2

$$f_{i,j}(x_i) = \sum_{o \in C} p_{i,j} * o(x_i) \tag{2}$$

where $o(x_i)$ is the operation of the output of ith node. An intermediate node is $x_j = \sum_{i<j}(f_{i,j}(x_i))$. The output of all the intermediate nodes is concatenated without the input nodes.

The search procedure is an alternative updating approach to optimize both architecture parameters and weight parameters. After search operation, the largest $d^{o}_{i,j}$ operation for a pair of nodes and the two connections with the largest $d^{o}_{i,j}$ for each intermediate node will be preserved.

3.2 PC-DARTS Overview

PC-DARTS is an improved DARTS by reducing the large memory consumption. It successfully utilize partial channel connection to increase the batch size and minimize the searching time, thus will lead to more stable results. Consider a partial connection case, which is sampled by a mask $\mathrm{S}_{i,j}$. This is depicted in the upper part of Fig. 1. i and j from s is the input and output node of each partial channel connection part. By this mask, the features are divided into two parts, the selected one, which will go through the candidate operations, and the masked one, which will be directly sent to the output node.

$$f^{PC}_{i,j}(\mathrm{x}_i; \mathrm{S}_{i,j}) = \sum_{o \in O} \frac{exp\left\{\alpha^{o}_{i,j}\right\}}{\sum_{o' \in O} exp\left\{\alpha^{o'}_{i,j}\right\}} \cdot o\left(\mathrm{S}_{i,j} * \mathrm{x}_i\right) + (1 - \mathrm{S}_{i,j}) * \mathrm{x}_i \tag{3}$$

where, the sampled channels $\mathrm{S}_{i,j} * \mathrm{x}_i$ and masked channels $(1 - \mathrm{S}_{i,j}) * \mathrm{x}_i$ are set to a proportion parameter K.

Edge normalization is another key improvement used for mitigating the undesired fluctuation caused by unbalanced sampling. The paper puts weights $\beta_{i,j}$ on each edge. Then the output node can be calculated as:

$$\mathrm{x}^{PC}_{j} = \sum_{i<j} \frac{exp\left\{\beta_{i,j}\right\}}{\sum_{i'<j} exp\left\{\beta_{i',j}\right\}} \cdot f_{i,j}\left(\mathrm{x}_i\right) \tag{4}$$

3.3 Batch Normalization for Architecture Search

We can diversify all cells no matter if it is reduction cell or a normal cell by introducing a scaling factor gamma for each channel. The overall loss function is show in Equation 5:

$$L = \sum_{(x,y)} l(f(x, W), y) + \lambda\left(\sum g(\gamma_o) + \sum g(\gamma_e)\right) \tag{5}$$

where, x denotes the input, y denotes the output, W denotes network weights, $g(\cdot)$ is L1 sparsity norm penalty function of search and unbalanced edge alleviate operation. λ is used to balance the two terms. By giving penalty of all the learnable gamma parameters, the sparsity of parameters will be improved and the important operation can be selected.

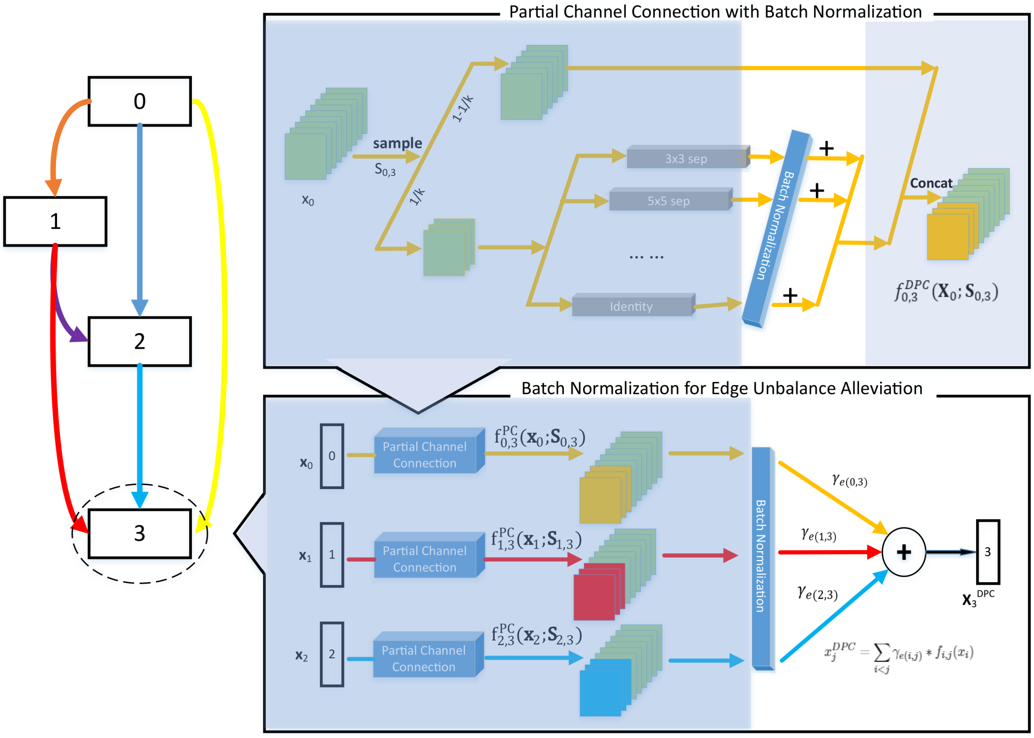

Figure 1: Illustration of AD-DARTS. To simplify the architecture parameters, we add the batch normalization after the optional operations which are based on 1/k of the samples. By using the scaling factor of batch normalization, the uncertainty of sampling can be reduced.

Upper part of Figure 1 describes the whole idea of batch normalization (Ioffe and Szegedy, 2015) for architecture search. A typical BN structure is shown as below, B_in and B_out can be input and output of a BN layer.

$$\widetilde{B} = \frac{B_{in} - \mu_l}{\sqrt{\sigma_l^2 + \epsilon}}; B_{out} = \gamma * \widetilde{B} + \beta \tag{6}$$

where μ_l and σ_l are the mean and standard deviation values of input activation in layer l, gamma and beta are trainable affine transformation parameters (scale and shift).

We assume that this batch normalization is after the different operations and before the concatenation of all channels. According the value of γ_o parameters, which is the $L1$ norm summation of each optional operation, the most important operation will be selected. In order to reduce the memory of the whole searching process, we choose partial connection between the input and output. The following function shows mathematical explanation of the idea,

$$f_{i,j}^{DPC}(x_i; C_{i,j}) = concat(\sum_{o \in C} o(C_{i,j} * x_i), (1 - C_{i,j}) * x_i) \tag{7}$$

where, $C_{i,j} * x_i$ and $(1 - C_{i,j}) * x_i$ denote the selected and masked channels, respectively. In experiment, K as a proportion is used to mask channels. The value is set following the paper (Xu et al., 2019).

3.4 Batch Normalization for Edge Unbalance Alleviation

Since the input channels go through the operations are randomly sampled during the iteration, there is an unbalance between sampled and masked channels. This could lead to network unstable. In order to solve this, we put the batch normalization behind the intermediate nodes, which is shown in the lower part of figure 1. It is depicted by the following function:

$$x_j^{DPC} = \sum_{i<j} \gamma_{e(i,j)} * f_{i,j}(x_i) \tag{8}$$

where, $\gamma_{e(i,j)}$ comes from the scaling factor of batch normalization for each pairs of points i and j.

4 Experiments and Implementation Details

4.1 Implementation Overview

We follow the DARTS way of sequential execution, which consist of two stages, namely searching and training. During the search stage, the best architecture is found according to the learned architecture parameters. After that, the training stage starts from scratch. Unlike DARTS, we don't need to separate train set into two sets, since there is no need to optimized architecture parameters any longer. To make it comparable, the operation space of AD-DARTS is the same as in DARTS, namely: 3×3 and 5×5 separable convolution, 3×3 and 5×5 dilated separable convolution, 3×3 max-pooling, 3×3 average pooling, skip-connect (identity), and zero (none).

References to the image are missing?

The network during searching stage consists of 10 cells (8 normal and 2 reduction cells), where each cell has 7 nodes, which are 2 input nodes, 4 intermediate nodes, and 1 output node respectively. We set the batch size to 160, number of epochs to 60 and the initial number of channels to 16. The network is optimized by momentum SGD (Polyak and Juditsky, 1992; Qian, 1999) with an initial learning rate of 0.1 (annealed down to zero following a cosine schedule without restart) (Loshchilov and Hutter, 2016), momentum 0.9, and weight decay 3×10^{-4}. The entire search takes only 4 hours on 2 $\times$ RTX2080 gpus with less than 16G memory.

The evaluation stage is different from the one proposed in original DARTS, because each cell is different. So, there is no change of the entire structure. The initial number of channels is set to 36, where the network is trained with 600 epochs using batch size of 64. We follow DARTS with initial learning rate of 0.025 (annealed down to zero following a cosine schedule without restart), momentum 0.9, weight decay 3×10^{-4}, norm gradient clipping at 5 and drop path with a rate of 0.3 as well as cutout (DeVries and Taylor, 2017). The only change compared to CIFAR100 is that we set the weight decay to 5×10^{-4}.

4.2 Results

The proposed approach has been tested on CIFAR10 and CIFAR100 datasets. Listed datasets contain natural images with resolution 32×32 (width $\times$ height) pixels. CIFAR10 has 10 classes and CIFAR100 has 100 classes respectively, where both sets consist of 50K training and 10K testing images.

Table 1 shows that DPC-DARTS can achieve comparable results with only half numbers of parameters and 1/7.5 searching time in the original first order DARTS. The proposed approach with 10 layers even obtain a better result compared to the first order DARTS.

Table 2 shows the results on CIFAR100. We both transfer the architectures searched on CIFAR10 to CIFAR100 and directly search on CIFAR100. It's easy to see the searched one is better than transferred one, although it consists little more parameters. The searched networks are shown in table 4 and fig. ??. In order to see the structure of cells, we visualize the first normal cell, fig. 2 and reduction cell ??.

Table 1: Comparison with state-of-the-art image classifiers on CIFAR10.

Architecture	Test Error (%)	Params (M)	Search Cost (GPU days)	#ops	Search Method
DenseNet-BC (Huang et al., 2017)	3.46	25.6	-	-	manual
NASNet-A + cutout (Zoph and Le, 2016)	2.65	3.3	2000	13	RL
BlockQNN (Zhong et al., 2018)	3.54	39.8	96	8	RL
AmoebaNet-A (Shah et al., 2018)	3.34	3.2	3150	19	evolution
AmoebaNet-B + cutout	2.55	2.8	3150	19	evolution
PNAS	3.41	3.2	255	8	SMBO
ENAS + cutout	2.89	4.6	0.5	6	RL
DARTS (first order) + cutout	3.00	3.3	1.5	7	gradient-based
DARTS (second order) + cutout	2.76	3.3	4	7	gradient-based
SNAS (mild) + cutout	2.98	2.9	1.5	7	gradient-based
SNAS (aggressive) + cutout	3.10	2.3	1.5	7	gradient-based
BayesNAS + cutout (Zhou et al., 2019)	2.81	3.4	0.2	7	gradient-based
PC-DARTS + cutout	2.57	3.6	0.1	7	gradient-based
DPC-DARTS + cutout (10 layers)	2.90	1.6	0.2	7	gradient-based
DPC-DARTS + cutout (9 layers)	3.08	1.4	0.2	7	gradient-based

Table 2: Comparison with state-of-the-art image classifiers on CIFAR100.

Architecture	Test Error (%)	Params (M)	Search Cost (GPU days)	#ops	Search Method
DARTS (first order) + cutout	17.76	3.3	1.5	7	gradient-based
DARTS (second order) + cutout	17.54	3.3	4.0	7	gradient-based
DPC-DARTS CIFAR10 + cutout (10 layers)	18.23	1.7	0.2	7	gradient-based
DPC-DARTS CIFAR100 + cutout (10 layers)	17.87	1.8	0.2	7	gradient-based

Table 3: 10 layers network architecture searched on CIFAR100, where column titles are start nodes x_i and row titles are end nodes x_j.

		middle0	middle1	middle2	middle3
layer1	input0	skip-connect	sep-conv-5x5	skip-connect	skip-connect
normal	input1	skip-connect	sep-conv-3x3		
cell	middle0			skip-connect	
	middle1				sep-conv-5x5
	middle2				
layer2	input0	skip-connect	sep-conv-3x3	sep-conv-5x5	sep-conv-5x5
normal	input1	sep-conv-3x3	sep-conv-3x3	sep-conv-3x3	sep-conv-5x5
cell	middle0				
	middle1				
	middle2				
layer3	input0	sep-conv-3x3	sep-conv-3x3	max-pool-3x3	
normal	input1	skip-connect	sep-conv-3x3		sep-conv-3x3
cell	middle0				
	middle1			max-pool-3x3	sep-conv-5x5
	middle2				
layer4	input0	max-pool-3x3	avg-pool-3x3		sep-conv-5x5
reduction	input1	max-pool-3x3		sep-conv-5x5	sep-conv-5x5
cell	middle0		max-pool-3x3		
	middle1			sep-conv-3x3	
	middle2				
layer5	input0	dil-conv-5x5	sep-conv-3x3	max-pool-3x3	sep-conv-3x3
normal	input1	sep-conv-3x3	max-pool-3x3		sep-conv-5x5
cell	middle0			max-pool-3x3	
	middle1				
	middle2				
layer6	input0	max-pool-3x3	sep-conv-3x3	sep-conv-3x3	sep-conv-3x3
normal	input1	dil-conv-5x5	sep-conv-5x5	max-pool-3x3	sep-conv-3x3
cell	middle0				
	middle1				
	middle2				
layer7	input0	sep-conv-5x5	sep-conv-5x5		sep-conv-5x5
reduction	input1	max-pool-3x3			
cell	middle0		max-pool-3x3	max-pool-3x3	
	middle1			dil-conv-5x5	
	middle2				dil-conv-5x5
layer8	input0	dil-conv-5x5	dil-conv-5x5		dil-conv-5x5
normal	input1	dil-conv-5x5	dil-conv-5x5	dil-conv-5x5	dil-conv-5x5
cell	middle0				
	middle1			sep-conv-5x5	
	middle2				
layer9	input0	max-pool-3x3	dil-conv-5x5	dil-conv-5x5	
normal	input1	dil-conv-5x5	sep-conv-3x3		dil-conv-5x5
cell	middle0			dil-conv-5x5	
	middle1				
	middle2				max-pool-3x3
layer10	input0	sep-conv-5x5	dil-conv-5x5	sep-conv-5x5	

normal	input1	sep-conv-5x5	sep-conv-5x5		
cell	middle0			sep-conv-5x5	
	middle1				max-pool-3x3
	middle2				max-pool-3x3

Table 4: 10 layers network architecture searched on CIFAR10, where column titles are start nodes x_i and row titles are end nodes x_j.

		middle0	middle1	middle2	middle3
layer1	input0	sep-conv-3x3	sep-conv-3x3	skip-connect	
normal	input1	max-pool-3x3	sep-conv-3x3		
cell	middle0			skip-connect	
	middle1				max-pool-3x3
	middle2				max-pool-3x3
layer2	input0	sep-conv-5x5	skip-connect	skip-connect	sep-conv-5x5
normal	input1	sep-conv-5x5	max-pool-3x3	sep-conv-3x3	
cell	middle0				sep-conv-3x3
	middle1				
	middle2				
layer3	input0	max-pool-3x3	sep-conv-3x3	sep-conv-3x3	
normal	input1	max-pool-3x3			
cell	middle0		sep-conv-3x3		sep-conv-5x5
	middle1			sep-conv-3x3	
	middle2				sep-conv-3x3
layer4	input0	sep-conv-5x5	sep-conv-5x5	max-pool-3x3	skip-connect
reduction	input1	max-pool-3x3	sep-conv-5x5	sep-conv-5x5	sep-conv-5x5
cell	middle0				
	middle1				
	middle2				
layer5	input0	sep-conv-5x5	sep-conv-5x5		
normal	input1	sep-conv-3x3	sep-conv-5x5	dil-conv-3x3	sep-conv-3x3
cell	middle0			dil-conv-3x3	sep-conv-3x3
	middle1				
	middle2				
layer6	input0	max-pool-3x3	max-pool-3x3	sep-conv-3x3	max-pool-3x3
normal	input1	max-pool-3x3	max-pool-3x3		avg-pool-3x3
cell	middle0			max-pool-3x3	
	middle1				
	middle2				
layer7	input0	sep-conv-5x5	sep-conv-5x5	avg-pool-3x3	sep-conv-5x5
reduction	input1	sep-conv-5x5			
cell	middle0		dil-conv-5x5		
	middle1			dil-conv-5x5	
	middle2				dil-conv-5x5
layer8	input0	dil-conv-5x5	dil-conv-5x5		sep-conv-5x5
normal	input1	dil-conv-5x5	dil-conv-5x5	dil-conv-5x5	dil-conv-5x5
cell	middle0			dil-conv-5x5	
	middle1				
	middle2				
layer9	input0	max-pool-3x3		dil-conv-5x5	

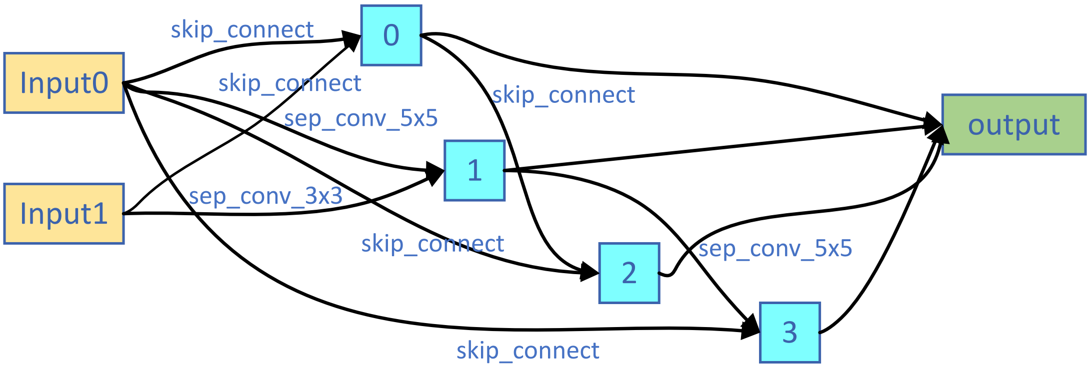

Figure 2: First reduction cell (layer 1) searched on CIFAR100.

normal cell	input1	dil-conv-5x5	dil-conv-5x5	sep-conv-5x5	dil-conv-5x5
	middle0		max-pool-3x3		
	middle1				
	middle2				max-pool-3x3
layer10 normal cell	input0	sep-conv-5x5			
	input1	sep-conv-5x5	max-pool-3x3		
	middle0		max-pool-3x3	sep-conv-5x5	max-pool-3x3
	middle1			max-pool-3x3	
	middle2				max-pool-3x3

5 Conclusions

We presented an improved DARTS, which is a simple, efficient, easy to implement and train approach named diversified partial connected differentiable architecture search (DPC-DARTS). By adding learnable scaling factors to the network, we extremely improved the diversity of original DARTS architecture. Consequently, the parameters are highly reduced which also means the inference time will be highly reduced. This is meaningful to be deployed in the embedded hardware for using it in the practical world. Another important contribution is by applying learnable scaling factors, we alleviate the unbalance of different channels and make the network much more stable.

References

Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. 2017. Smash: one-shot model architecture search through hypernetworks. arXiv preprint arXiv:1708.05344.

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

Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. 2019. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. arXiv preprint arXiv:1904.12760.

Terrance DeVries and Graham W Taylor. 2017. Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:1708.04552.

Jiemin Fang, Yuzhu Sun, Qian Zhang, Yuan Li, Wenyu Liu, and Xinggang Wang. 2019. Densely connected search space for more flexible neural architecture search. arXiv preprint arXiv:1906.09607.

Xinyu Gong, Shiyu Chang, Yifan Jiang, and Zhangyang Wang. 2019. Autogan: Neural architecture search for generative adversarial networks. In Proceedings of the IEEE International Conference on Computer Vision, pages 3224–3234.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778.

Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. 2019. Searching for mobilenetv3. arXiv preprint arXiv:1905.02244.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 4700–4708.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167.

Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. Technical report, Citeseer.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105.

Hanwen Liang, Shifeng Zhang, Jiacheng Sun, Xingqiu He, Weiran Huang, Kechen Zhuang, and Zhenguo Li. 2019. Darts+: Improved differentiable architecture search with early stopping. arXiv preprint arXiv:1909.06035.

Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. 2017. Learning efficient convolutional networks through network slimming. In Proceedings of the IEEE International Conference on Computer Vision, pages 2736–2744.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2018. Darts: Differentiable architecture search. arXiv preprint arXiv:1806.09055.

Ilya Loshchilov and Frank Hutter. 2016. Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983.

Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. 2018. Efficient neural architecture search via parameter sharing. arXiv preprint arXiv:1802.03268.

Boris T Polyak and Anatoli B Juditsky. 1992. Acceleration of stochastic approximation by averaging. SIAM Journal on Control and Optimization, 30(4):838–855.

Ning Qian. 1999. On the momentum term in gradient descent learning algorithms. Neural networks, 12(1):145–151.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. 2019. Regularized evolution for image classifier architecture search. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 4780–4789.

Syed Asif Raza Shah, Wenji Wu, Qiming Lu, Liang Zhang, Sajith Sasidharan, Phil DeMar, Chin Guok, John Macauley, Eric Pouyoul, Jin Kim, et al. 2018. Amoebanet: An sdn-enabled network service for big data science. Journal of Network and Computer Applications, 119:70–82.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Dimitrios Stamoulis, Ruizhou Ding, Di Wang, Dimitrios Lymberopoulos, Bodhi Priyantha, Jie Liu, and Diana Marculescu. Single-path nas: Designing hardware-efficient convnets in less than 4 hours.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1–9.

Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. 2019. Mnasnet: Platform-aware neural architecture search for mobile. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2820–2828.

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

Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. 2019. Pc-darts: Partial channel connections for memory-efficient differentiable architecture search. arXiv preprint arXiv:1907.05737.

Xiawu Zheng, Rongrong Ji, Lang Tang, Baochang Zhang, Jianzhuang Liu, and Qi Tian. 2019. Multinomial distribution learning for effective neural architecture search.

Zhao Zhong, Zichen Yang, Boyang Deng, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. 2018. Blockqnn: Efficient block-wise neural network architecture generation. arXiv preprint arXiv:1808.05584.

Hongpeng Zhou, Minghao Yang, Jun Wang, and Wei Pan. 2019. Bayesnas: A bayesian approach for neural architecture search. arXiv preprint arXiv:1905.04919.

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

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. 2018. Learning transferable architectures for scalable image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 8697–8710.