Operation-level Progressive Differentiable Architecture Search

Xunyu Zhu
Institute of Information Engineering
Chinese Academy of Sciences
Beijing, China
zhuxunyu@iie.ac.cn

Jian Li*
Institute of Information Engineering
Chinese Academy of Sciences
Beijing, China
lijian9026@iie.ac.cn

Yong Liu
Gaoling School of Artificial Intelligence
Renmin University of China
Beijing, China
liuyonggsai@ruc.edu.cn

Jun Liao
China Unicom Research Institute
China Unicom
Beijing, China
LIAOJ@chinaunicom.cn

Weiping Wang
Institute of Information Engineering
Chinese Academy of Sciences
Beijing, China
wangweiping@iie.ac.cn

***Abstract*—Differentiable Neural Architecture Search (DARTS) is becoming more and more popular among Neural Architecture Search (NAS) methods because of its high search efficiency and low compute cost. However, the stability of DARTS is very inferior, especially skip connections aggregation that leads to performance collapse. Though existing methods leverage Hessian eigenvalues to alleviate skip connections aggregation, they make DARTS unable to explore architectures with better performance. In the paper, we propose operation-level progressive differentiable neural architecture search (OPP-DARTS) to avoid skip connections aggregation and explore better architectures simultaneously. We first divide the search process into several stages during the search phase and increase candidate operations into the search space progressively at the beginning of each stage. It can effectively alleviate the unfair competition between operations during the search phase of DARTS by offsetting the inherent unfair advantage of the skip connection over other operations. Besides, to keep the competition between operations relatively fair and select the operation from the candidate operations set that makes training loss of the supernet largest. The experiment results indicate that our method is effective and efficient. Our method's performance on CIFAR-10 is superior to the architecture found by standard DARTS, and the transferability of our method also surpasses standard DARTS. We further demonstrate the robustness of our method on three simple search spaces, i.e., S2, S3, S4, and the results show us that our method is more robust than standard DARTS.**

***Index Terms*—DARTS, Neural Architecture Search, skip connections aggregation, search space**

I. Introduction

The development of machine learning [1]–[3] and deep learning [4]–[6] has promoted the revolution in the field of image classification. However, the cost of neural network architectures designed by hand is very high because it needs experienced experts to design high-performance network architectures, which means that network architecture design is unable to common for people. The problem impedes the development of deep learning. Fortunately, the emergence of neural architecture search (NAS) effectively alleviates the problem. It can make neural network design automatically and design high-quality networks that can be par with manually designed architectures.

Architecture search methods based on reinforcement learning and evolutionary algorithms are the first architecture search methods to be proposed. [7] proposes to leverage reinforcement learning to sample subnetworks and make the validation loss as the reward to help NAS select the best sub-architecture. [8] proposes to leverage evolutionary algorithms to explore subnetworks with better performance. However, these methods based on reinforcement learning and evolutionary algorithms cost are cumbersome. NASNet [9] proposes that NAS can select cells instead of the entire networks and stack cells to build networks. The method reduces computational consumption by compressing search space.

This paper proposes operation-level progressive differentiable architecture search (OPP-DARTS) to alleviate skip connections aggregation. Figure 1 shows the basic procedure of OPP-DARTS. We first divide the search phase into several stages. In the first stage, we select one candidate operation with parameters to join in the supernet. After the first operation joined in the supernet, we train the supernet several epochs until the next stage. In the next stage, we select another operation to join the supernet and train the supernet. We repeat this action at every stage. Then we train the supernet until the final epoch. In addition, different operations are unfair to compete with each other during the training phase, such as $3 \times 3\_sep\_conv$ and $skip\_connect$. DARTS always would like to select $skip\_connect$ rather than $3 \times 3\_sep\_conv$, which means when $skip\_connect$ joins the supernet earlier than other operations, the supernet prefers to select $skip\_connect$. Therefore, we make the operation loss as the criteria to help the supernet select corresponding operation, i.e., the candidate operation that makes operation loss largest will be selected to join in the network.

*Corresponding author.

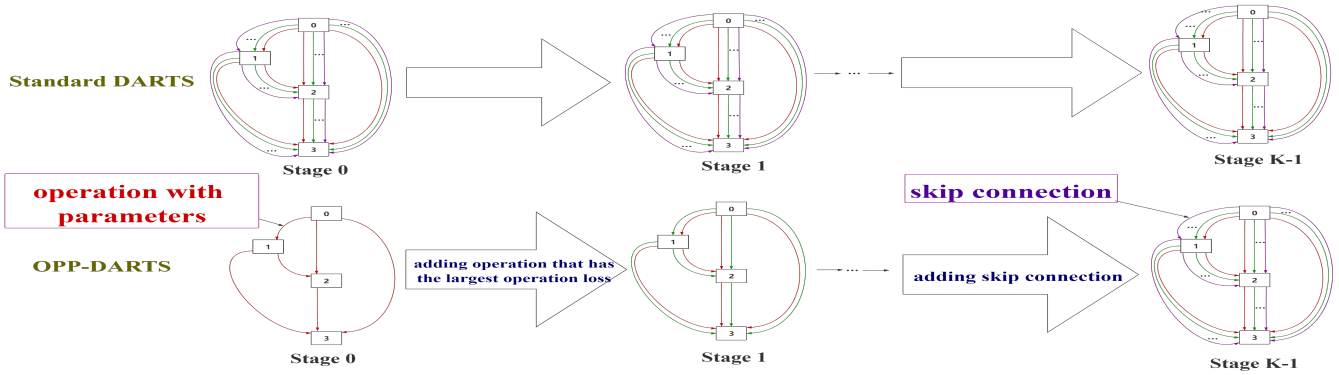


Fig. 1: The bottom figure is an overview of OPP-DARTS and the above figure shows the search process of standard DARTS.

OPP-DARTS has demonstrated its effectiveness and efficiency in Experiments (Section IV). We firstly search architecture on CIFAR-10 by using similar training settings with DARTS. Then, we evaluate the searched architecture on CIFAR-10 and ImageNet, and the training settings are also the same with DARTS. When evaluating it on CIFAR-10, the test error of the searched architecture is 2.63% with 3.8M parameters, and the performance is much better than standard DARTS. To verify the transferability of OPP-DARTS, we transfer the searched architecture to train on ImageNet, and the performance achieved on ImageNet shows that our method outperforms standard DARTS. At last, we further demonstrate the robustness of our method. We ran our method on three simple search spaces, i.e., **S2**, **S3**, **S4**, and the final results indicate that the robustness of our method is also better than standard DARTS.

II. Related Work

DARTS is a great innovation that relaxes neural architecture search into a continuous problem, i.e., makes NAS differentiable. Furthermore, this dramatically reduces computation cost and can search architecture with good performance in a single GPU day. However, DARTS has an unavoidable problem, and the problem prevents DARTS from being applied. The problem is skip connections aggregation, i.e., the architectures searched by DARTS are filled with excessive skip connections, and this phenomenon makes DARTS performance collapse. Some previous works have dealt with the problem. [11] proposes Hessian eigenvalues as a signal to monitor skip connections aggregation. SDARTS [12] regularizes Hessian eigenvalues by increasing perturbations to alleviate skip connections aggregation. P-DARTS [13] draws up rules manually to alleviate skip connections aggregation. Existing methods regard Hessian eigenvalues or the number of skip connections as the signal to guide to alleviate skip connections aggregation. These methods make some rules by hand or regularize Hessian eigenvalues to keep the stability of DARTS. However, these methods seek quick success and instant benefits because they make DARTS unable to explore architectures with better performance. DARTS- [14] adds an auxiliary skip connection to alleviate skip connections aggregation. The method is simple, whereas memory consumption will increase because of the extra skip connection.

III. Method

A. Preliminary of DARTS

Cell-based NAS methods [9], [15], [16] have been proposed to learn cells instead of architectures to reduce computation overhead in the process of architecture search. The networks consist of many same cell structures. A cell is similar to a directed acyclic graph (DAG), and it contains N nodes, i.e., two input nodes, some intermediate nodes, and a single output node. Each node is a latent representation symbolized as $x^{(i)}$, and each directed edge denoted as (i,j) represents an information flow that an operation $o^{(i,j)}$ in the directed edge (i,j) transfers information from node $x^{(i)}$ to node $x^{(j)}$. Differentiable Architecture Search (DARTS) [17] relaxes the selection problem of operations as a continuous optimization problem. There is a candidate operations space $\mathcal{O}$, where $o \in \mathcal{O}$ represents a candidate operation, e.g., $skip\_connect, 3\times 3\_sep\_conv$, and so on. In DARTS [17], each edge consists of a set of operations from $\mathcal{O}$, and these operations are weighted by architecture parameters $\alpha^{i,j}$, and it can be formulated as:

$$\bar{o}^{(i,j)}(x) = \sum_{o\in\mathcal{O}} \frac{\exp\left(\alpha_o^{(i,j)}\right)}{\sum_{o'\in\mathcal{O}}\exp\left(\alpha_{o'}^{(i,j)}\right)} o(x), \tag{1}$$

where $0 \le i < j \le N-1$. The input nodes of the cell take the output from the previous two cells as input, and the output node contacts all intermediate nodes as the output of the cell. Each intermediate node is be obtained by its predecessors, i.e. $x^{(j)} = \sum_{i<j} o^{(i,j)}\left(x^{(i)}\right)$. Finally, the architecture search of DARTS becomes a bi-level optimization problem:

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

where $\mathcal{L}_{val}$ and $\mathcal{L}_{\text{train}}$ are validation and training loss, w is the network weights, and α is the architecture weights. According to DARTS [17], the bi-level optimization problem is solved by

a first/second-order approximation. When the search process is nearly finished, an optimal substructure is obtained based on the architecture weights α, i.e., $o^{(i,j)} = \text{argmax}_{o \in \mathcal{O}} \alpha_o^{(i,j)}$.

B. Increasing Operations Progressively

To eliminate unfairness between operations, we propose operation-level progressive differentiable architecture search, briefly called "OPP-DARTS". In other words, we increase operations progressively to expand the search space during the search phase of DARTS, as illustrated in Figure 1, and the OPP-DARTS is detailed in Alg. 1. Firstly, the search phase of DARTS is divided into several stages denoted as $stage\ 1...K$, each stage consists of T epochs, and one operation will be increased into the search space of DARTS every stage.

Then, we will begin increasing operations to enlarge search space. Because the supernet must own an operation with parameters to make itself trainable, we must select an operation with parameters at $stage\ 0$. We first define candidate operations with parameters set as $\mathcal{O}_{pm}$. Furthermore, we select an operation o from candidate operations set $\mathcal{O}_{pm}$. Then, we increase it into the supernet and begin to train the supernet until the next stage.

Afterwards, we will define the candidate operations set as $\mathcal{O}$, and the candidate operations set $\mathcal{O}$ includes all candidate operations except the operation selected in $stage\ 0$ and $skip\_connect$. At the beginning of each stage, we select an operation from candidate operations set $\mathcal{O}$, and increase it into the supernet to train, i.e.,

$$\min_{\alpha} \quad \mathcal{L}_{val}\left(o, \Omega, w^*, \alpha\right) \tag{2}$$

$$\text{s.t.} \quad w^* = \text{argmin}_w \, \mathcal{L}_{\text{train}}\left(o, \Omega, w, \alpha\right) \tag{3}$$

$$o \in \mathcal{O}, \tag{4}$$

where o is the operation selected at the stage and Ω is the search space of the supernet. In this way, these operations increased into the supernet early have the advantage compared to those increased into the supernet lately because these operations that are increased into the supernet early can learn more valuable knowledge than those operations. We can alleviate the unfair natural advantages between operations to make operations compete fairly. When we pick operations at random from the candidate operations set $\mathcal{O}$, it may cause a negative effect if we select operations with natural advantages compared with the others to increase it into the supernet earlier than other operations. Thus, we need to design a criterion to guide us to select operations, and Section III-C shows a criterion to alleviate the problem.

In $stage\ K-1$, we increase $skip\_connect$ into the supernet because skip connection has the most significant natural advantages than other operations. By increasing it into the supernet, at last, we can make other operations offset the natural advantages of skip connection to make operations compete fairly.

C. Operation Loss

By increasing operations progressively, we can make operations compete fairly during the search phase. However, the problem still needs to be solved, i.e., when we select operations from the candidate operations set $\mathcal{O}$ at each stage, we need to decide to select which operation to increase in the supernet. The problem is very critical because if we select operations with natural advantages compared with the others to increase it into the supernet earlier than other operations, it will negatively impact it. Thus, we need to design an applicable criterion to guide us to make operation selection.

[21] indicates that edges share the same optimal feature map in a cell, and it means the edge feature graphs will be closer and closer as the network converges. The feature map of an edge can be represented as Eq. 1. At initialization, if the feature map on an operation is close to the optimal feature map, the architecture parameter α of the operation will become larger at the beginning of the architecture search. It will result in unfairness in initialization. To keep operations competing fairly, we select the operation which can make the supernet gain the largest training loss after the operation is increased into the supernet at the beginning of each stage, called "operation loss", i.e.,

$$o = argmax_{o \in \mathcal{O}} \mathcal{L}_{train}(o, w, \alpha), \tag{5}$$

where o is the operation that is selected at the stage. When we use "operation loss" as a criterion to guide us to select an operation at the beginning of each stage, we can make feature maps of operations relatively close to the optimal feature map before they begin to compete with each other. We increase the operation that owns the largest "operation loss" to train at each stage so that the operation can be more closed to the optimal feature map. Thus, it can alleviate unfairness in initialization. Furthermore, the search process of DARTS can be formalized as follows:

$$\min_{\alpha} \quad \mathcal{L}_{val}\left(o, \Omega, w^*, \alpha\right) \tag{6}$$

$$\text{s.t.} \quad w^* = \text{argmin}_w \, \mathcal{L}_{\text{train}}\left(o, \Omega, w, \alpha\right) \tag{7}$$

$$o = argmax_{o \in \mathcal{O}} \mathcal{L}_{train}(o, \Omega, w, \alpha), \tag{8}$$

where Ω is the search space of the supernet. In addition, the feature map of operation is saved in the parameters of operation. Training loss is used to optimize network parameters, i.e., training loss has a maximum correlation with the feature map of operation, so we select training loss instead of validation loss.

We know that skip connection owns a natural advantage because it can make the network coverage faster, and it means that skip connection owns a more significant advantage than other operations, so we increased it into the supernet at the final stage to keep operations compete fairly.

D. Discussion

In the paper, our method is proposed to deal with performance collapse in DARTS arising from skip connections aggregation. Our method owns two contributions mainly, i.e., operation-level increasing operations progressively and operation loss. The first contribution is a very significant innovation to alleviate skip connections aggregation. Compared with

Algorithm 1 Operation-level Progressive Differentiable Architecture Search (OPP-DARTS)

Require: The search space of the supernet Ω, all candidate operations set $\Re$.
Select $\vartheta \in \mathcal{O}_{pm}$ based on Eq. (5) to join in Ω;
Initialize $\mathcal{O} = \Re \setminus \{skip\_connect, \vartheta\}$
for $i = 1$ *to* E **do**
  if $i \ \% \ T == 0$ and $i \leq T \times k$ **then**
    Select $o \in \mathcal{O}$ based on Eq. (5) to join in Ω;
    Remove o from O;
  end if
  if $i == T \times (k+1)$ **then**
    Select $skip\_connect$ to join in M;
  end if
  Update architecture parameters α by descending $\bigtriangledown_{\alpha}\mathcal{L}_{val}(w, \alpha)$;
  Update weights w by descending $\bigtriangledown_{w}\mathcal{L}_{train}(w, \alpha)$;
end for

other works to alleviate performance collapse in DARTS by indicator-based methods (e.g., R-DARTS [11], SDARTS [12]), we alleviate performance collapse by optimizing the search space of DARTS to keep operations compete with each other fairly. Thus the view of our research is very novel. The second contribution is proposed to solve the operation selection problem at the beginning of each stage. If we select operations from candidate operations set at random, operations will suffer from unfairness in initiation, making DARTS work worse. By operation loss, we can alleviate unfairness in the initiation and make DARTS work well. We obtain a good performance (a test error of 2.63%) on CIFAR-10 by combining two contributions, and the result indicates that our method makes a remarkable improvement in accuracy on standard DARTS. At the same time, the transferability and robustness of our method are also demonstrated to be better than standard DARTS.

IV. Experiments

We first search normal and reduction cells on CIFAR-10 [22] in Section IV-A . Then, we stack the selected cells to build a new deeper network to evaluate its performance on CIFAR-10 in Section IV-B. Further, Section IV-C shows the transferability of OPP-DARTS by stacking cells to build an even deeper network and then evaluating it on ImageNet. Finally, we verify the robustness of our methods by searching architectures on some simple search space, i.e., **S2, S3, S4** in Section IV-D.

A. Architecture Search on CIFAR-10

CIFAR-10 is an images dataset that contains $50K$ training images and $10K$ test images. In the next moment, we will introduce our search space. Our search space consists of eight operation, i.e., $max\_pool\_3 \times 3, avg\_pool\_3 \times 3, zero, sep\_conv\_3 \times 3, sep\_conv\_5 \times 5, dil\_conv\_3 \times 3, dil\_conv\_5 \times 5$, it is the same with DARTS. Our training settings are also the same with DARTS, we stack 6 normal cells and 2 reduction cells to build a network, and the reduction cells are inserted in a network of $1/3$ and $2/3$, respectively. A cell includes 7 nodes with 4 intermediate nodes, every node represents a feature map, and the number of edges in a cell is 14. The input of two input nodes in a cell is the output of two previous cells, and the output node of a cell is the concatenation of all intermediate nodes.

TABLE I: Comparison with state-of-the-art image classifiers on CIFAR-10 (lower error rate is better).

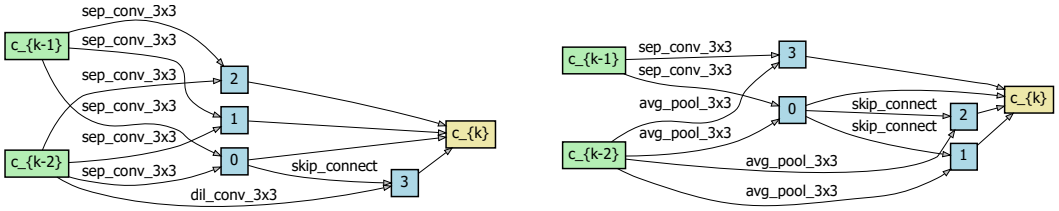
Architecture	Test Err. (%)	Params (M)	Search Cost (GPU-days)	Search Method
DenseNet-BC [25]	3.46	25.6	-	manual
NASNet-A [9]	2.65	3.3	1800	RL
AmoebaNet-A [16]	3.34±0.06	3.2	3150	evolution
AmoebaNet-B [16]	2.55±0.05	2.8	3150	evolution
PNAS [15]	3.41±0.09	3.2	225	SMBO
ENAS [10]	2.89	4.6	0.5	RL
DARTS (1st order) [17]	3.00±0.14	3.3	0.4	gradient
DARTS (2nd order) [17]	2.76±0.09	3.3	1	gradient
SNAS (mild) [18]	2.98	2.9	1.5	gradient
ProxylessNAS [26]	2.08	-	4	gradient
P-DARTS [13]	2.5	3.4	0.3	gradient
PC-DARTS [19]	2.57±0.07	3.6	0.1	gradient
SDARTS-RS [12]	2.67±0.03	3.4	0.4	gradient
GDAS [27]	2.93	3.4	0.3	gradient
R-DARTS (L2) [11]	2.95±0.21	-	1.6	gradient
SGAS (Cri 1. avg) [28]	2.66±0.24	3.7	0.25	gradient
DARTS-PT [21]	2.61±0.08	3.0	0.8	gradient
OPP-DARTS	2.63±0.27 [1]	3.8	0.4 [2]	gradient

[1] We ran OPP-DARTS 5 times with different search seeds to search cells, and evaluated the best cells 10 times with different evaluation seeds to get average test error and variance of test error.
[2] Recorded on a single GTX 2080Ti.

We perform an architecture search for 50 epochs with a batch size of 64, and the architecture search is divided into 8 stages. Each stage includes 2 epochs. We increase one candidate operation in the search space at the beginning of each stage and train the supernet until the next stage. After the final stage, we train the supernet until the final epoch. During the training, we leverage SGD [23] as an optimizer to optimize model weights W, and its initial learning rate is 3×10^{-4}, momentum is $(0.5, 0.999)$, weight decay is 10^{-3}. At the same time, we leverage Adam [24] to optimize architecture weights, and its initial learning rate is 3×10^{-4}, momentum is $(0.5, 0.999)$, weight decay is 10^{-3}. The search phase spends 0.4 GPU day on a single NVIDIA GTX 2080Ti. We ran 5 times independent search experiments with 5 different seeds, and the best cells were found are illustrated in Figure 2.

B. Architecture Evaluation on CIFAR-10

The cells found by OPP-DARTS have shown in Figure 2, and then we will stack these cells to evaluate their performance on CIFAR-10. We build a large network with 20 cells and 36 initial channels. After building the network, we train the stacked network until 600 epochs with a batch size of 96. Furthermore, other training settings are the same with DARTS, such as cutout with length 16, auxiliary towers with weight 0.4, and path dropout with a probability of 0.3.



(a) Normal cell (b) Reduction cell

Fig. 2: Best cells are found by OPP-DARTS on CIFAR-10 (a). Normal cell is found by OPP-DARTS. (b). Reduction cell is found by OPP-DARTS.

We compare our result on CIFAR-10 with other methods in Table I, such as Darts- [14], SGAS [28]. Table I shows that our method (OPP-DARTS) improves standard DARTS' test error from 3.00% to 2.63%, with identical search cost (0.4 GPU days) on a single NVIDIA GTX 2080Ti. Our method surpasses them a lot compared to other indicator-based variants, such as SDARTS-RS and R-DARTS (L2). The result further demonstrates that the indicator-based method can alleviate skip connections aggregation but prevent DARTS from exploring better architectures.

C. Architecture Evaluation on ImageNet

To verify the transferability of our method, we evaluate architecture on ImageNet by using the best cells searched on CIFAR-10. ILSVRC 2012 [29] is a famous ImageNet dataset, and we leverage it to test our architecture searched on CIFAR-10.

The network configuration is also the same as DARTS, i.e., the network used to evaluate is stacked by 14 cells, and the number of its initial channels is 48. The stacked network is trained from scratch for 250 epochs, and its batch size is 256 during the training phase of the network. An SGD optimizer optimizes the network's parameters; the optimizer's initial learning rate is 0.2, weight decay is 3×10^{-5}, momentum is 0.9.

Table II shows our evaluation result, and we compare our result with SOTA manual architectures and models obtained through other search methods. The architecture found by OPP-DARTS on CIFAR-10 is superior to the architecture found by standard DARTS in the image classification task, and it means that the transferability of our method is better than standard DARTS.

D. Robustness of OPP-DARTS

Because of skip connections aggregation, DARTS will face a performance crash when DARTS searches architectures on three simple search space [11], i.e., **S2**, **S3**, **S4**. **S2** is consist of two operations per edge: $(skip\_connect, 3 \times 3\_sep\_conv)$. **S3** is consist of three operations per edge: $(skip\_connect, 3 \times 3\_sep\_conv, zero)$. **S4** is consist of two operations per edge: $(noise, 3 \times 3\_sep\_conv)$. [11] discovers that when DARTS is used to search architectures on **S2**, **S3**, it finds suboptimal architectures, and the performance of architectures is inferior. Even though on **S4**, DARTS selects excessive harmful $noise$ operations.

TABLE II: Comparison with state-of-the-art classifiers on ImageNet.

Architecture	Test Err.(%)		Params	$\times+$	Search Cost	Search
	top-1	top-5	(M)	(M)	(GPU-days)	Method
Inception-v1 [30]	30.2	10.1	6.6	1448	-	manual
MobileNet [31]	29.4	10.5	4.2	569	-	manual
ShuffleNet 2x (v1) [32]	26.4	10.2	$\sim$5	524	-	manual
ShuffleNet 2x (v2) [33]	25.1	-	$\sim$5	591	-	manual
NASNet-A [9]	26	8.4	5.3	564	1800	RL
NASNet-B [9]	27.2	8.7	5.3	488	1800	RL
NASNet-C [9]	27.5	9	4.9	558	1800	RL
AmoebaNet-A [16]	25.5	8	5.1	555	3150	evolution
AmoebaNet-B [16]	26	8.5	5.3	555	3150	evolution
AmoebaNet-C [16]	24.3	7.6	6.4	570	3150	evolution
FairNAS-A [34]	24.7	7.6	4.6	388	12	evolution
PNAS [15]	25.8	8.1	5.1	588	225	SMBO
MnasNet-92 [35]	25.2	8	4.4	388	-	RL
DARTS(2^{nd} order) [17]	26.7	8.7	4.7	574	4.0	gradient
SNAS (mild) [18]	27.3	9.2	4.3	522	1.5	gradient
ProxylessNAS [26]	24.9	7.5	7.1	465	8.3	gradient
P-DARTS [13]	24.4	7.4	4.9	557	0.3	gradient
PC-DARTS [19]	25.1	7.8	5.3	586	0.1	gradient
SGAS (Cri.1 avg.) [28]	24.41	7.29	5.3	579	0.25	gradient
GDAS [27]	26.0	8.5	5.3	581	0.21	gradient
OPP-DARTS	25.61	8.15	5.3	610	0.4	gradient

TABLE III: OPP-DARTS on **S2**-**S4** (test error ($\%$))

Space	DARTS	OPP-DARTS
S2	5.94	2.93
S3	3.04	2.90
S4	4.85	3.21

To verify the robustness of our methods, we search architectures on these search spaces by OPP-DARTS. Table III shows the results of the OPP-DARTS search in the **S2**-**S4**. When searching on **S2**, the performance of OPP-DARTS (a test error of 2.84%) is far further than DARTS. At the same time, we use OPP-DARTS to search architectures on **S3** and **S4**. Table III also shows that the performance of OPP-DARTS on **S3** and **S4** is better than DARTS. Based on these above experiments, OPP-DARTS is more robust than DARTS.

V. Conclusions

In this paper, we propose operation-level progressive differentiable architecture search to alleviate skip connections aggregation. The core idea is that we split the search space into multiple stages and increase operations into search space at the beginning of each stage. In addition, we select the operation that makes the training loss of the supernet largest

from candidate operations set to keep operations competing fairly.

VI. Acknowledgements

This work is supported in part by Excellent Talents Program of Institute of Information Engineering, CAS, Special Research Assistant Project of CAS (No. E0YY231114), Beijing Outstanding Young Scientist Program (No. BJJWZYJH012019100020098) and National Natural Science Foundation of China (No. 62076234, No. 62106257). At the same time, the work is also supported by Intelligent Social Governance Platform, Major Innovation & Planning Interdisciplinary Platform for the "Double-First Class" Initiative, Renmin University of China. We also wish to acknowledge the support provided and contribution made by Public Policy and Decision-making Research Lab of Renmin University of China.

References

[1] J. Li, Y. Liu, and W. Wang, "Automated spectral kernel learning," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 2020, pp. 4618–4625. [Online]. Available: https://aaai.org/ojs/index.php/AAAI/article/view/5892

[2] J. Li, Y. Liu, R. Yin, and W. Wang, "Multi-class learning using unlabeled samples: Theory and algorithm," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, S. Kraus, Ed. ijcai.org, 2019, pp. 2880–2886. [Online]. Available: https://doi.org/10.24963/ijcai.2019/399

[3] R. Yin, Y. Liu, W. Wang, and D. Meng, "Distributed nyström kernel learning with communications," in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 2021, pp. 12 019–12 028. [Online]. Available: http://proceedings.mlr.press/v139/yin21a.html

[4] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012.

[6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[7] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," 2017. [Online]. Available: https://arxiv.org/abs/1611.01578

[8] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," 2019.

[9] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning Transferable Architectures for Scalable Image Recognition," *arXiv e-prints*, p. arXiv:1707.07012, Jul. 2017.

[10] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," in *ICML*, 2018.

[11] A. Zela, T. Elsken, T. Saikia, Y. Marrakchi, T. Brox, and F. Hutter, "Understanding and robustifying differentiable architecture search," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=H1gDNyrKDS

[12] X. Chen and C.-J. Hsieh, "Stabilizing differentiable architecture search via perturbation-based regularization," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 1554–1565. [Online]. Available: http://proceedings.mlr.press/v119/chen20f.html

[13] X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive differentiable architecture search: Bridging the depth gap between search and evaluation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[14] X. Chu, X. Wang, B. Zhang, S. Lu, X. Wei, and J. Yan, "{DARTS}-: Robustly stepping out of performance collapse without indicators," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=KLH36ELmwIB

[15] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive Neural Architecture Search," *arXiv e-prints*, p. arXiv:1712.00559, Dec. 2017.

[16] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized Evolution for Image Classifier Architecture Search," *arXiv e-prints*, p. arXiv:1802.01548, Feb. 2018.

[17] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable Architecture Search," *arXiv e-prints*, p. arXiv:1806.09055, Jun. 2018.

[18] S. Xie, H. Zheng, C. Liu, and L. Lin, "SNAS: Stochastic Neural Architecture Search," *arXiv e-prints*, p. arXiv:1812.09926, Dec. 2018.

[19] Y. Xu, L. Xie, X. Zhang, X. Chen, G.-J. Qi, Q. Tian, and H. Xiong, "Pc-darts: Partial channel connections for memory-efficient architecture search," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=BJlS634tPr

[20] X. Chu, T. Zhou, B. Zhang, and J. Li, "Fair darts: Eliminating unfair advantages in differentiable architecture search," in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham: Springer International Publishing, 2020, pp. 465–480.

[21] R. Wang, M. Cheng, X. Chen, X. Tang, and C.-J. Hsieh, "Rethinking architecture selection in differentiable NAS," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=PKubaeJkw3

[22] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[23] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural Networks*, vol. 12, no. 1, pp. 145–151, 1999.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[25] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2261–2269.

[26] H. Cai, L. Zhu, and S. Han, "ProxylessNAS: Direct neural architecture search on target task and hardware," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=HylVB3AqYm

[27] X. Dong and Y. Yang, "Searching for a robust neural architecture in four gpu hours," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 1761–1770.

[28] G. Li, G. Qian, I. C. Delgadillo, M. Müller, A. Thabet, and B. Ghanem, "Sgas: Sequential greedy architecture search," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[29] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[30] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 2014.

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

[32] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6848–6856.

[33] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design," in *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham: Springer International Publishing, 2018, pp. 122–138.

[34] X. Chu, B. Zhang, R. Xu, and J. Li, "Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search," 2020.

[35] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, jun 2019, pp. 2815–2823. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR.2019.00293