Decoupled Differentiable Neural Architecture Search: Memory-Efficient Differentiable NAS via Disentangled Search Space

Libin Hou [1] **Linyuan Wang** [1] **Bin Yan** [1]

Abstract

Differentiable Neural Architecture Search (NAS) is a popular paradigm, but scaling this approach to models with larger parameters is severely hampered by the fact that the entire supernet resides in GPU memory. In this paper, we rethink the gradient propagation processs of Differentiable NAS and propose Decoupled Differentiable Neural Architecture Search (D2NAS). In our method, the branch structure is designed to decouple the weight update of the trainable parameters from the backbone network, and the candidate operation selection is redesigned with Gumbel-Softmax to make the overall differentiable process more stable. Experiments show that D2NAS achieves both performance and stability, with 67% memory cost compared to the best other differentiable methods.

1. Introduction

Neural Architecture Search (NAS) [1] has made rapid progress in neural network design for vision tasks such as classification [2], [3], object detection [4] and segmentation [5]. Early approaches based on evaluating a large number of candidate models required an unaffordable cost [6], [7]. Inspired by the weight sharing mechanism, various low-cost approaches have emerged [8]. DARTS [9] introduces architecture weights to indicate the importance of each operation and employs first-order and second-order approximations to update operation parameters and architecture weights through stochastic gradient descent. DARTS dominates with its fast speed and has a lot of follow-up research [3], [10]–[13].

The scalability is essential since larger search spaces hold more high quality models, as depicted by the increasing average accuracy of models with increasing search space sizes. However, the comprehensive nature of the supernet, encompassing all connections and operations in the search space, poses a challenge of excessive GPU memory consumption. There is an urgent need for larger-scale models but the shortening of the search process does not necessarily mean an equivalent reduction in GPU memory usage [14]. Therefore, designing a new method to effectively reduce GPU memory usage is crucial to fully exploit the utility of Differentiable NAS.

In this paper, we revisit the gradient process of search and point out that *the intermediate gradient of caching activation values in previous methods becomes a significant part of GPU memory footprint*. Therefore, Decoupled Differentiable Neural Architecture Search (D2NAS) is proposed to replace the above design of entanglement. The Disentangled Search Space containing Branch Cell is designed, see Fig. 1 for details. *Only Branch cells are updated during the search to avoid a large accumulation of gradients of Normal cells*. Moreover, we use Gumbel-Softmax to make the selection of candidate operations more stable than previous methods [10], [11].

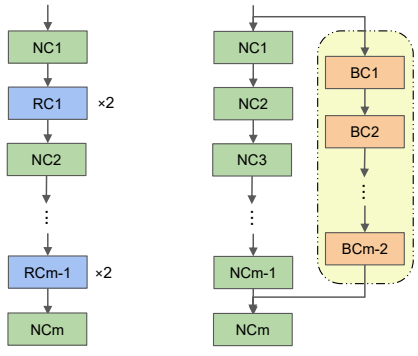

Figure 1. Illustration of the differences of neural architecture between D2NAS and previous approaches. (a) Vanilla Search Space from [9] is stacked with Normal Cell (NC) and Reduction Cell (RC). The whole architecture bears the burden of all gradient propagation while searching. (b) Our proposed Disentangled Search Space is composed of Normal Cell (NC) and Branch Cell (BC). Only the architecture parameters of BC are updated during the search.

[1]Henan Key Laboratory of Imaging and Intelligence Processing, Zhengzhou, China. Correspondence to: Libin Hou <lbhou98@163.com>.

Published at the 2^{nd} Differentiable Almost Everything Workshop at the 41^{st} International Conference on Machine Learning, Vienna, Austria. July 2024.

2. Related work

In order to reduce GPU memory cost, some previous studies modified the forward process of the supernet.

DARTS-like Methods. P-DARTS [15] proposes the search space approximation strategy, which reduces the number of candidate operations accordingly whenever the number of layers increases. Reducing the number of candidate operations frees up the GPU memory footprint, but also affects the accuracy of the search. PC-DARTS [11] employs partial connections instead of a full supernet.

Single-path Methods. Certain research works have suggested consolidating all parameterized operations into a unified convolution, with a comparable hyperkernel strategy applied in single-path NAS studies. ProxylessNAS [10] samples two operations per edge during the search, which enables agent-free search on large datasets. Single-path supernet like SPOS [16] and FairNAS [17] only sample one path per iteration, however, both require additional search phases to select the final model. Single-path differentiable methods like GDAS [18] sample the subgraph of the Directed acyclic graphs(DAG) in each iteration, which is by far the most efficient method.

However, the iteration of the differentiable process itself consumes a lot of GPU memory, and this problem is not mitigated, which has been neglected in previous studies. In our method D2NAS, the differentiable search procedure is optimized from the decoupled space.

3. Preliminary

The Differentiable NAS framework is defined as follows. Given a specific task, such as a classification task, with training set D_{train} and validation set D_{val}, a unit can be represented as a Directed Acyclic Graph (DAG) with N nodes, each node representing a potential feature. Each directed edge (i, j) is associated with an operation, $o(i, j) \in \mathcal{O}$, where $\mathcal{O}$ is a set of candidate operations. DARTS [9] relaxes the learnable architecture parameters α continuously to blend the outputs of operations as

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

Candidate architecture is an N layer convolution neural network $y = f_N(f_{N-1}(...f_1(x)))$, where layer i has a weight matrix W_i and a bias term b_i. We denote o_{i+1}, z_{i+1} as the output and pre-activation of layer i, respectively. Then, $o_{i+1} = \sigma(z_{i+1}) = \sigma(W_i o_i + b_i)$, where σ is the activation function.

At the search stage, NAS can be modeled as a bi-level optimization problem that minimizes the loss L by alternately updating the operation weights w (the parameters of the candidate operations on each edge) and the architecture parameters α,

$$\min_{\alpha} \mathcal{L}_{\text{val}}\ (w^*(\alpha), \alpha) \tag{2}$$

$$\text{s.t. } w^*(\alpha) = \arg\min_{w} \mathcal{L}_{\text{train}}\ (w, \alpha^*)\,. \tag{3}$$

The gradients back propagated from the loss L to W_i and b_i are

$$\begin{aligned} \frac{\partial L}{\partial W_i} &= \frac{\partial L}{\partial o_{i+1}} \sigma_i' o_i\,, \\ \frac{\partial L}{\partial b_i} &= \frac{\partial L}{\partial o_{i+1}} \sigma_i'\,, \end{aligned} \tag{4}$$

where σ_i' is the abbreviation of $\partial o_{i+1}/\partial z_{i+1}$. Furthermore, the term $\partial L/\partial o_{i+1}$ can be recursively expressed as

$$\frac{\partial L}{\partial o_{i+1}} = \frac{\partial L}{\partial o_{i+2}} \frac{\partial o_{i+2}}{\partial z_{i+2}} \frac{\partial z_{i+2}}{\partial o_{i+1}} = \frac{\partial L}{\partial o_{i+2}} \sigma_{i+1}' W_{i+1}\,. \tag{5}$$

To correctly calculate the gradients, except for parameters from the model (in this case, W_i and b_i), all corresponding $\{\sigma_i'\}$ in the chain rule have to be cached during search, which dominates the GPU memory usage.

4. Method

The gradient update is consistent with the Eq. 2 in Disentangled Search Space, but the gradient merging of Normal Cell and Branch Cell is prone to produce gradient collapse in the computational accumulation. In order to improve the stability of the method, instead of Eq. 1, we propose to use Gumbel softmax for the selection of candidate operations, which is mainly described below:

To disentangle the search for topology and operations on each edge, we use an indicator $B_{i,j} \in \{0, 1\}$ to denote whether edge $e_{i,j}$ is selected, and $A_{i,j}^o \in \{0, 1\}$ for whether operation o on edge $e_{i,j}$ is selected. Sampling architecture z with M connections can be decomposed into two parts: sample M edges first, and their operations second.

Sampling for edges. Topology inconsistency exists in single-path based methods [19], [20], as all 14 edges in a cell are selected in the search stage but the final architecture only has 8 edges. To address this issue, we propose to sample the same number of edges in search.

Each intermediate node should connect with exact two predecessors, satisfying the constraint of DARTS. Formally, we use $B_{i,j}$ to indicate whether the edge $e_{i,j}$ between node x_i and x_j is sampled, and we enforce,

$$\sum_{i<j} B_{i,j} = 2, \quad \forall j. \tag{6}$$

Sampling for operations. We use $A_{i,j}^o$ to denote whether the operator o is sampled on the edge $e_{i,j}$, and we adopt

Gumbel-Softmax technique to sample operations, where $g^o_{i,j}$ is sampled from Gumbel$(0,1)$ distribution and $g^o_{i,j} = -\log(-\log(\epsilon^o_{i,j}))$, $\epsilon^o_{i,j}$ obeys uniform distribution, and $\tilde{\alpha}^o_{i,j} = \frac{\exp(\alpha^o_{i,j})}{\sum_{o' \in \mathcal{O}} \exp(\alpha^{o'}_{i,j})}$ is the normalized architectural weights:

$$A_{i,j} = \mathcal{O}_{\text{one\_hot}} \left[\arg\max_{o \in \mathcal{O}} (\log \tilde{\alpha}^o_{i,j} + g^o_{i,j}) \right], \quad (7)$$

To make the objective function differentiable to architectural weights α, we relax the discrete distribution to a continuous one by Gumbel-Softmax:

$$\tilde{A}^o_{i,j} = \frac{\exp\left[(\log \tilde{\alpha}^o_{i,j} + g^o_{i,j})/\tau\right]}{\sum_{o'=1}^{|\mathcal{O}|} \exp\left[(\log \tilde{\alpha}^{o'}_{i,j} + g^{o'}_{i,j})/\tau\right]},$$
$$A_{i,j} = \mathcal{O}_{\text{one\_hot}} \left[\arg\max_{o \in \mathcal{O}} \tilde{A}^o_{i,j} \right], \quad (8)$$

where the temperature τ gradually decreases in the search process.

5. Experiment

5.1. Settings

Search space. The search space of DARTS is denoted as S0, which includes a stack of duplicate normal cells and reduction cells. Each cell is represented by a DAG with four intermediate nodes. The candidate operations between every two nodes are {maxpool, avgpool, skip_connect, sep_conv 3×3 and 5×5, dil_conv 3×3 and 5×5}.We search and evaluate on CIFAR-10 [21] and ImageNet [22] under S0 search space, respectively.
We also perform experiments on the four reduced search spaces, S1-S4, introduced by R-DARTS [23] to assess the stability of our approach. S1 represents a pre-optimized search space where each edge in the supernet is associated with a predefined set of candidate operations. In the remaining three search spaces, the candidate operations on each edge remain consistent. Consistent with R-DARTS [23], the search and evaluation are conducted within these four search spaces using CIFAR-10, CIFAR-100 [21], and SVHN [24] datasets.

Search Settings Similar to DARTS, the supernet comprises 8 cells with 16 initial channels. The search is conducted over 50 epochs with a sampling number set to $K = 7$. The operation parameters are optimized using the SGD optimizer with a momentum of 0.9 and an initial learning rate of 0.05. As for the architectural weights, the Adam optimizer is employed with an initial learning rate of 3×10^{-4}.

Evaluation Settings The evaluation settings mirror those of DARTS [9], where the inferred model is trained for 600 epochs using SGD with a batch size of 96. In the search space S0, inferred models are built by stacking 20 cells with 36 initial channels and trained under the same conditions as outlined in [9], [15]. For S1-S4, we adhere strictly to the settings established in R-DARTS [23] to ensure a fair comparison.

5.2. Results

Performance in S0 on CIFAR. We follow DARTS [9] and search on the CIFAR-10. Table 1 shows that achieves state-of-the-art performance with only 0.3 GPU-days. D2NAS has an average of 2.58±0.07% error rate, which is slightly higher than up-to-date SOTAs such as SDARTS-ADV [25]. However, D2NAS is more than 4× faster. Compared with R-DARTS [23], D2NAS robustly outperforms it with 5× fewer search costs. Our best model achieves 97.52% accuracy with 3.6M parameters.

Models	**Params** (M)	**FLOPs** (M)	**Error** (%)	**Cost** (GPU Days)	SP
DARTS-V1 [9]	3.3	528	3.38±0.23	0.4	×
P-DARTS [15]‡	3.3±0.2	540±34	2.81±0.14	0.3	×
PC-DARTS [26]‡	3.6±0.5	592±90	2.89±0.22	0.1	×
PR-DARTS [27]‡	3.4	-	2.68±0.10	0.2	×
R-DARTS [23]	-	-	2.95±0.21	1.6	×
SDARTS-ADV [25]	3.3	-	2.61±0.02	1.3	×
ZARTS [28]	3.7	-	2.54±0.07	1.0	×
Few-shot NAS [29]	3.8	-	2.31±0.08	1.35	×
GDAS [19]	3.4	-	2.93	0.2	✓
SNAS [20]	2.8	-	2.85±0.02	1.5	✓
D2NAS (best)	3.5	593	2.29	0.3	
D2NAS (avg.)	3.7±0.4	595±25	2.31±0.08	0.3	

Table 1. Averaged performance among 4 independent runs of search on CIFAR-10. ‡: reproduced result using their released code since they didn't report the average performance. †: FLOPs are calculated by their released architecture. SP: single-path based method

We also search on CIFAR-100 and show the results in Table 2. D2NAS surpasses all the methods and achieves state-of-the-art with only 0.3 GPU-days search cost.

Performance in S0 on ImageNet. First, we transfer the architecture searched on CIFAR-10 to ImageNet following the common practice [9], [15], [17]. Same as [17], we train models for 250 epochs with a batch size of 1024 by SGD optimizer with a momentum of 0.9 and an initial lr of 0.5 base learning rate. We also utilize an auxiliary classifier strategy. The results are shown in Table 3, where D2NAS achieves 75.3% top-1 accuracy.
Second, as D2NAS features low memory cost and great robustness, we directly search on ImageNet as well. We randomly sample 10% images to train operation parameters and another 10% to train architectural weights. A supernet is constructed by stacking 8 cells with 16 initial channels.

Models	Params (M)	Error (%)	Cost (GPU Days)
ResNet [30]	1.7	22.10$^{\diamond}$	-
AmoebaNet [31]	3.1	18.93$^{\diamond}$	3150
PNAS [32]	3.2	19.53$^{\diamond}$	150
ENAS [33]	4.6	19.43$^{\diamond}$	0.45
DARTS [9]	-	20.58±0.44*	0.4
GDAS [19]	3.4	18.38	0.2
P-DARTS [15]	3.6	17.49‡	0.3
R-DARTS [23]	-	18.01±0.26	1.6
ZARTS [28]	4.1	16.29±0.53	1.0
D2NAS (best)	3.2	17.39	0.3
D2NAS (avg.)	3.4±0.5	17.52±0.13	0.3

Table 2. Comparison of searched models on CIFAR-100. $^{\diamond}$: Reported by [19], *: Reported by [23], ‡:Rerun their code.

Models	FLOPs (M)	Params (M)	Top-1 (%)	Cost (GPU days)
AmoebaNet-A [31]	555	5.1	74.5	3150
NASNet-A [34]	564	5.3	74.0	2000
PNAS [35]	588	5.1	74.2	225
DARTS [9]	574	4.7	73.3	0.4
P-DARTS [15]	577	5.1	75.3	0.3
FairDARTS-B [17]	541	4.8	75.1	0.4
SNAS [20]	522	4.3	72.7	1.5
PC-DARTS [26]	586	5.3	74.9	0.1
GDAS [19]	581	5.3	74.0	0.2
D2NAS (ours)	576	5.2	75.3	0.3
DARTS, P-DARTS	OOM‡ when batch-size ≥ 32			
FairNAS-C [17]	321	4.4	74.7	12
SPOS [16]	323	3.5	74.4	12
ProxylessNAS [10]	465	7.1	75.1	8.3
FBNet-C [36]	375	5.5	74.9	9
PC-DARTS [26]	597	5.3	75.4	3.8
GDAS [19]	405	3.6	72.5	0.8
D2NAS (ours)	592	4.8	75.8	0.4

Table 3. Performance on ImageNet. The first block indicates the models *transferred* from CIFAR-10; The second block indicates that the models are *directly searched* on ImageNet.
‡OOM: Out of Memory while running codes on GPU.

We search for 30 epochs with $K = 3$. The batch size is set as 256. Our search cost is reduced to 0.4 GPU days on a single Tesla V100. We fully train the discovered model for 250 epochs with the same evaluation settings as above. Results are illustrated in Tabel 3, showing that D2NAS achieves 75.8% top-1 accuracy. To make fair comparisons, we reproduce GDAS [19] under the same settings (90 epochs). However, the network is dominated by skip connections and only achieves 72.5% top-1 accuracy.

Robustness Evaluation. We follow the recommended best practices for NAS by [23], [25] to report the mean and variance across several times of parallel searching with various random seeds, through which the robustness of a method can be measured.
We follow R-DARTS [23] to evaluate the performance and across 3 datasets in S1-S4 search spaces, see Table 5(Appendix). Our methods robustly outperform other methods with a clear margin across all these benchmarks.

Memory Analysis. Table 4 compares GPU memory cost in S0 search space on CIFAR-10. D2NAS has the lowest memory cost thanks to our disentanglement of the search for topology. Unlike GDAS that preserves multiple edges for each node, we strictly sample 2 edges for each node leading to 26% memory reduction compared to GDAS.
PC-DARTS [26] uses partial channels during the search stage to reduce GPU memory cost, in which the partial ratio is controlled by a hyperparameter M. But M requires careful calibration for different tasks. In contrast, D2NAS doesn't require calibrating such an extra hyperparameter and is more memory-efficient.

Method	DARTS	GDAS	PC-DARTS		D2NAS
			M=4	M=2	
Memory (G)	9.4	3.1	3.7	5.7	2.1

Table 4. GPU Memory cost comparison. We measured the cost based on a batch size of 64, where the supernet has 16 initial channels, and 8 layers.

6. Conclusion

This study introduces a Memory-Efficient method with Disentangled Search Space in Differentiable Neural Architecture Search (NAS). We show that the decoupled search space in the differentiable search framework can search good performance models while saving memory resources. Experimental results on CIFAR10, CIFAR-100, SVHN and ImageNet datasets reveal that D2NAS approach achieves competitive performance compared to other differentiable methods, with only about 67% of their GPU memory usage and more stability.

References

[1] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *CVPR*, 2018.

[2] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the*

International Conference on Machine Learning (ICML), PMLR, 2019, pp. 6105–6114.

[3] X. Chu, X. Wang, B. Zhang, S. Lu, X. Wei, and J. Yan, "Darts-: Robustly stepping out of performance collapse without indicators," en, in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[4] G. Ghiasi, T.-Y. Lin, and Q. V. Le, "Nas-fpn: Learning scalable feature pyramid architecture for object detection," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[5] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. L. Yuille, and L. Fei-Fei, "Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[6] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017. [Online]. Available: https://arxiv.org/pdf/1611.01578.pdf.

[7] M. Tan, "Mnasnet: Platform-aware neural architecture search for mobile," en, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[8] G. Bender, P.-J. Kindermans, B. Zoph, V. Vasudevan, and Q. Le, "Understanding and simplifying one-shot architecture search," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

[9] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," en, in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[10] H. Cai, L. Zhu, and S. Han, "Proxylessnas: Direct neural architecture search on target task and hardware," *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[11] 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*, 2019.

[12] H. Liang, *Darts+: Improved differentiable architecture search with early stopping*, en, arXiv preprint arXiv:1909.06035, 2019.

[13] P. Ye, B. Li, Y. Li, T. Chen, J. Fan, and W. Ouyang, "Beta-darts: Beta-decay regularization for differentiable architecture search," en, in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, 10864–10873,

[14] R. Wang, M. Cheng, X. Chen, X. Tang, and C.-J. Hsieh, "Rethinking architecture selection in differentiable nas," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[15] 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*, 2019, pp. 1294–1303.

[16] Z. Guo, X. Zhang, H. Mu, W. Heng, Z. Liu, Y. Wei, and J. Sun, "Single path one-shot neural architecture search with uniform sampling," in *European Conference on Computer Vision*, Springer, 2020, pp. 544–560.

[17] X. Chu, T. Zhou, B. Zhang, and J. Li, "Fair darts: Eliminating unfair advantages in differentiable architecture search," in *European conference on computer vision*, Springer, 2020, pp. 465–480.

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

[19] ——, "Searching for a robust neural architecture in four gpu hours," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[20] S. Xie, H. Zheng, C. Liu, and L. Lin, "Snas: Stochastic neural architecture search," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[21] Tech. Rep.

[22] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[23] A. Zela, T. Elsken, T. Saikia, Y. Marrakchi, T. Brox, and F. Hutter, "Understanding and robustifying differentiable architecture search," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[24] Y. Netzer and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2011.

[25] X. Chen and C.-J. Hsieh, "Stabilizing differentiable architecture search via perturbation-based regularization," in *Proceedings of the International Conference on Machine Learning (ICML)*, PMLR, 2020, pp. 1554–1565.

[26] 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 *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[27] X. C. Zhou Pan, "Theory-inspired path-regularized differential network architecture search," in *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[28] X. Wang, W. Guo, J. Su, X. Yang, and J. Yan, "Zarts: On zero-order optimization for neural architecture search," in *Proceedings of the 36th International Conference on Neural Information Processing Systems*, ser. NIPS '22, ¡conf-loc¿, ¡city¿New Orleans¡/city¿, ¡state¿LA¡/state¿, ¡country¿USA¡/country¿, ¡/conf-loc¿: Curran Associates Inc., 2024, ISBN: 9781713871088.

[29] Y. Zhao, L. Wang, Y. Tian, R. Fonseca, and T. Guo, "Few-shot neural architecture search," in *Proceedings of the International Conference on Machine Learning (ICML)*, PMLR, 2021, pp. 12 707–12 718.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[31] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[32] 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," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 19–34.

[33] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

[34] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[35] 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," in *ECCV*, 2018.

[36] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer, "Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

A. Appendix

A.1. The comparative experiment for the stability of D2NAS

Benchmark		DARTS†	ES†	ADA†	GDAS		D2NAS	
		Error (%)	Error (%)	Error (%)	Error (%)	Num	Error(%)	Num
C10	S1	4.66±0.71	3.05±0.07	3.03±0.08	2.89±0.09	3.8±0.4	2.66±0.05	1.3±0.4
	S2	4.42±0.40	3.41±0.14	3.59±0.31	3.89±0.17	6.0±0.7	3.12±0.13	2.0±0.0
	S3	4.12±0.85	3.71±1.14	2.99±0.34	3.04±0.10	6.5±0.5	2.65±0.01	2.0±0.0
	S4	6.95±0.18	4.17±0.21	3.89±0.67	3.34±0.10	0.0±0.0	3.20±0.14	0.0±0.0
C100	S1	29.93±0.41	28.90±0.81	24.94±0.81	24.49±0.08	4.0±0.0	22.70±0.67	2.3±0.4
	S2	28.75±0.92	24.68±1.43	26.88±1.11	24.57±0.47	6.3±0.4	22.87±0.74	3.5±0.5
	S3	29.01±0.24	26.99±1.79	24.55±0.63	22.86±0.17	3.0±0.7	22.41±0.32	2.5±0.5
	S4	24.77±1.51	23.90±2.01	23.66±0.90	24.14±0.89	2.3±1.1	20.90±0.43	0.0±0.0
SVHN	S1	9.88±5.50	2.80±0.09	2.59±0.07	2.48±0.04	2.8±0.4	2.32±0.06	0.8±0.4
	S2	3.69±0.12	2.68±0.18	2.79±0.22	3.05±0.02	7.8±0.4	2.43±0.06	1.0±0.0
	S3	4.00±1.01	2.78±0.29	2.58±0.07	3.62±0.36	7.5±0.5	2.56±0.05	1.5±0.5
	S4	2.90±0.02	2.55±0.15	2.52±0.06	2.51±0.06	1.5±1.5	2.32±0.01	0.0±0.0

Table 5. Comparison in 4 search spaces and 3 datasets. For each algorithm, we independently search for 3 times under the settings in R-DARTS [23] and report the averaged performance. 'EA' and 'ADA' are two methods proposed by R-DARTS. 'Num': To reveal the collapse issue, we also report the average number of parameterless operations in the discovered normal cell for GDAS and D2NAS †: Results are obtained from R-DARTS.