OPT2021: 13th Annual Workshop on Optimization for Machine Learning

Simulated Annealing for Neural Architecture Search

Shentong Mo* SHENTONM@ANDREW.CMU.EDU
Jingfei Xia* JINGFEIX@ANDREW.CMU.EDU
Pinxu Ren PREN@ANDREW.CMU.EDU
Carnegie Mellon University, Pittsburgh, PA 15213, United States

Abstract

Gradient-based Neural Architecture Search (NAS) approaches have achieved remarkable progress in the automated machine learning community. However, previous methods would cause much search time and huge computation resources in a big search space for seeking an optimal network structure. In this work, we propose a novel Simulated Annealing algorithm for NAS, namely **SA-NAS**, by adding perturbations to the gradient-descent for saving search cost and boosting the predictive performance of the search architecture. Our proposed algorithm is easy to be adapted to current state-of-the-art methods in the literature. We conduct extensive experiments on various benchmarks where the results demonstrate the effectiveness and efficiency of our SA-NAS in reducing search time and saving computation resources. Compared to previous differentiable methods, our SA-NAS achieves comparable or better predictive performance under the same setting.

1. Introduction

Recently, Neural Architecture Search (NAS) has gained popularity in the automated machine learning (AutoML) community. The goal of NAS is that given the constraints of network size and latency, we need to design an efficient optimization algorithm to explore an optimal network structure in a large space of architectures. Most early approaches develop this optimal solution by applying reinforcement learning or evolution methods, but those methods require huge search costs and computation resources. To avoid searching in a large space of candidate architectures, DARTS [15] relax the search space to be continuous so that gradient descent can be utilized to optimize the architecture with respect to its validation set performance. PC-DARTS [20] further reduces the redundancy in the search space by using partial channel connections in a small part of the supernetwork, so that a more efficient architecture search algorithm is applied without a performance drop.

However, previous methods [3, 4, 13, 15, 20] need to take much search time and huge computation resources in a big search space for finding the optimal architecture, which tends to fall into saddle points. In the meanwhile, those gradient-based searching methods with random initialization [6, 8] can be significantly slowed down by saddle points since it would take *exponential* time to escape saddle points. Moreover, gradient descent [7, 9] with perturbations would not be slowed down by saddle points such that it can find an approximate local minimizer in *polynomial* time. The reason why we adopt gradient descent in the deep learning community most of the time is that gradient descent is an "economic" option when dealing with a large number of parameters. However, in the neural architecture search problem, the number of architecture parameters α in the normal and

*. These authors contributed equally to this work.

reduction cell is limited since the layers and blocks are predefined manually, *e.g.*, skip connection, dilated convolution 5×5 and average pooling 3×3. Therefore, a more effective method is urgently needed to speed up the neural architecture search.

Motivated by commonly-used simulated annealing algorithms [5], we propose a novel simulated annealing optimization algorithm for neural architecture search, termed SA-NAS, to save computation resources and reduce search time. Specifically, at each optimization iteration k, we simultaneously search the current architecture α^k and a random neighbouring architecture α_n^k on the validation set, and compare their separate objective $\mathcal{L}_{\text{val}}(w^*(\alpha^k), \alpha^k)$ and $\mathcal{L}_{\text{val}}(w^*(\alpha_n^k), \alpha_n^k)$. If $\mathcal{L}_{\text{val}}(w^*(\alpha^k), \alpha^k) > \mathcal{L}_{\text{val}}(w^*(\alpha_n^k), \alpha_n^k)$, the optimum value at current step i is α_n^k; otherwise, we take a certain probability p to accept α_n^k as the optimum architecture parameters for next step $k+1$. The probability p decreases with the iteration k such that the searching process becomes stable if the objective reaches the optimum value.

To further escape an oversimplified solution collapsing to many skip connections [3, 4, 13], we propose a simple yet effective regularizer $\sum_i \alpha_i^{\text{skip connect}}$ to the original objective $\mathcal{L}_{\text{val}}(w^*(\alpha), \alpha)$ to form our new objective. In the meantime, we also apply an early stopping mechanism if more than one skip connection layer in the normal cell is searched in the space. Extensive experiments on various benchmarks show that our SA-NAS achieves comparable and even better performance in terms of downstream tasks. Compared to state-of-the-art methods, the results also demonstrate the efficiency of our SA-NAS in reducing the optimized architecture's parameters and search timing cost.

Overall, our contributions in this work are summarized as follows:

- We propose a Simulated Annealing algorithm for NAS, namely SA-NAS, which can be easily adapted to previous gradient-based methods for reducing search cost.
- We introduce a simple yet effective regularizer in the objective to avoid an oversimplified solution collapsing to many skip connections.
- We conduct extensive experiments on various benchmarks where the results show the efficiency of our SA-NAS in saving computation resources and reducing search time.
- Our SA-NAS achieves comparable or better predictive performance, compared to current gradient-based approaches.

2. Related Works

2.1. Neural Architecture Search

Neural Architecture Search (NAS) [2, 11, 17–19, 22] has achieved remarkable progress in recent years. Generally, there are three main directions for methods in the NAS literature: including Reinforcement Learning based [17, 21, 22], evolutionary learning based [14, 18], and gradient descent based [3, 4, 13, 15, 20]. In this work, we mainly focus on differentiable approaches for neural architecture search. DARTS [15] is the first approach that applies gradient descent for effective architecture search, where the discrete architecture search space is relaxed to continuous differentiable search space for convolutional and cyclic architectures. Following up on DARTS, Chen *et al.* [3] present a progressively differentiable network architecture search, which makes the network depth in the search situation as close as possible to the network depth in the evaluation. To save

computational memory and cost, the channel sampling and edge normalization method are proposed in PC-DARTS [20]. To mitigate the deteriorative effect of many skip connections on the performance, DARTS+ [13] introduces an early stopping mechanism to reduce the search time and increase the predictive accuracy. Recently, Fair-DARTS [4], adds an auxiliary loss to the original objective to solve the discretization discrepancy existing in DARTS. In this work, we focus on ameliorating gradient-based methods and introducing a novel optimization algorithm that can be easily adapted to previous gradient-based methods in the NAS literature.

2.2. Simulated Annealing

Simulated annealing (SA) algorithm [5] is a probabilistic method to approach the global optimum for a given objective. The main idea for SA is that to pick random neighbors of the current best point and to evaluate their objectives. If the objective is smaller than its current best cost, then the neighboring point is accepted, and the best cost is updated. If the cost is higher than the best objective, then the point is accepted based on a predefined probability density of the Bolzman-Gibbs distribution. Recently, simulated annealing algorithms have been applied to many optimization problems. Particularly, MOSTASA [10] presents a novel state-transition simulated annealing algorithm for constrained and unconstrained multi-objective optimization problems. HHOBSA [1] combines SA with bitwise operations to propose a hybrid version of the Harris Hawks optimization algorithm for the feature selection problem on classification tasks. In this work, we apply the simulated annealing algorithm to NAS for seeking the optimal architecture to achieve the best predictive performance.

3. Method

In this section, we first introduce preliminaries about gradient-based methods. Then we present a novel algorithm for neural architecture search, namely SA-NAS, which can be easily adapted to previous gradient-based work. Finally, we propose a penalty term in the objective to escape an oversimplified solution collapsing to many skip connections.

3.1. Preliminary

The goal of the gradient-based method for NAS is to find a building block of the final structure, *i.e.*, cell. A cell is a directed acyclic graph composed of N nodes, where each node $x^{(i)}$ represents a feature map, and each edge $\mathcal{E}$ represents the operation of between $x^{(i)}$ and $x^{(j)}$. Each cell has two input edges and one output edge. The output of the cell is obtained by performing reduction operations, *e.g.*, concatenation, on all intermediate nodes. Let $\mathcal{O}$ denotes a set of candidate operations, and each operation is denoted by $o(\cdot)$. To make the search space continuous, the selection of a specific operation is relaxed as:

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

where $\alpha_{o^{(i,j)}}$ represents the score of an operation $o^{(i,j)}$ between node $x^{(i)}$ and $x^{(j)}$. Therefore, the goal of the neural architecture search is to learn the architecture α and the weights w within all the mixed operations (*e.g.* weights of the convolution filters). Then the bilevel optimization problem can

be defined as

$$\begin{aligned} &\min_{\alpha} \mathcal{L}_{\text{val}}(w^*(\alpha), \alpha) \\ &\text{s.t. } w^*(\alpha) =_w \mathcal{L}_{\text{train}}(w, \alpha) \end{aligned} \tag{2}$$

where $\mathcal{L}_{\text{train}}$ and $\mathcal{L}_{\text{val}}$ denote the training and the validation loss, respectively. Thus, the optimization algorithm is composed of two stage: first, we optimize the weights w^* associated with the architecture by minimizing the training loss $w^*(\alpha) =_w \mathcal{L}_{\text{train}}(w, \alpha)$; second, we minimize the validation loss $\mathcal{L}_{\text{val}}(w^*, \alpha)$ to find the optimal architecture α^*.

3.2. SA-NAS

Inspired by simulated annealing algorithms [5] for a random search, we propose a novel optimization algorithm called SA-NAS, for solving the problem defined in Eq. 2. Specifically, at each optimization iteration k, we search a random architecture α^k and a random neighbouring architecture α_n^k at the same time on the validation set, and compare their separate objective $\mathcal{L}_{\text{val}}(w^*(\alpha^k), \alpha^k)$ and $\mathcal{L}_{\text{val}}(w^*(\alpha_n^k), \alpha_n^k)$. If $\mathcal{L}_{\text{val}}(w^*(\alpha^k), \alpha^k) > \mathcal{L}_{\text{val}}(w^*(\alpha_n^k), \alpha_n^k)$, the optimum value at current step i is α_n^k; otherwise, we take a certain probability p to accept α_n^k as the optimum architecture parameters for next step $k+1$. The randomly-sampled neighbouring architecture α_n^k is defined as

$$\alpha_n^k = \alpha^k + \mathcal{N}(\mathbf{0}, \mathbf{I}), n \in \{1, 2, \cdots, N\} \tag{3}$$

where $\mathcal{N}(\mathbf{0}, \mathbf{I})$ is a standard normal distribution of the same shape as α^k, and N denote the total number of randomly neighboring architectures. And the probability p is defined as

$$p = \exp([-(\mathcal{L}_{\text{val}}(w^*(\alpha_n^k), \alpha_n^k) - \mathcal{L}_{\text{val}}(w^*(\alpha^k), \alpha^k)/(b * T)]) \tag{4}$$

where b, T denote the Boltzmann's constant and temperature parameter, respectively. Note that the probability p decreases with the iteration k such that the searching process becomes stable if the objective reaches the optimum value. This algorithm is easy to be adapted to previous gradient-based methods. Algorithm 1 shows the overall algorithm for SA-NAS.

3.3. Skip Connection Regularizer and Early Stopping

Previous work [3, 4, 13] have pointed out that solutions to Eq. 2 would collapse to a large number of skip connections as the searching iteration increases. To escape such an oversimplified solution of α, we propose a simple yet effective regularizer $\sum_i \alpha_i^{\text{skip connect}}$, namely Skip Connection Regularizer (SCR), to the original objective $\mathcal{L}_{\text{val}}(w^*(\alpha), \alpha)$ in Eq. 2 to form a new objective defined as

$$\begin{aligned} &\min_{\alpha} \mathcal{L}_{\text{val}}\left(w^*(\alpha), \alpha\right) + \lambda \sum_{o^{(i,j)}} \alpha_{o^{(i,j)}}^{\text{skip connect}} \\ &\text{s.t. } \; w^*(\alpha) = \arg\min_{w} \mathcal{L}_{\text{train}}\left(w, \alpha\right). \end{aligned} \tag{5}$$

where λ is a hyper-parameter to control the punishment scalar of the sum of αs for the skip connection operator. Thus, $\nabla\alpha$, *i.e.*, the gradient of the objective w.r.t. α is updated as

$$\nabla\alpha = \frac{\partial \mathcal{L}_{\text{val}}\left(w^*(\alpha), \alpha\right)}{\partial \alpha} + \lambda \mathbb{1} o^{(i,j)} \tag{6}$$

where $\mathbb{1}o^{(i,j)}$ is an indicator function, *i.e.*, if the operator $o^{(i,j)}$ is skip connection, $\mathbb{1}o^{(i,j)} = 1$; otherwise it is 0. In this way, we add a λ penalty term to the original gradient for the skip connection. With the penalty term, the mode is more "serious" choosing skip connections and will effectively lower the number of skip connections in the model. Inspired by DARTS+ [13], we also adopt an early stopping mechanism if the number of skip connection layers in the normal cell is more than s in the searched architecture. In this way, soft and hard punishments are imposed on the weights α of skip connection such that the optimized architecture achieves better performance.

Algorithm 1: Simulated annealing algorithm for SA-NAS

1. Initialize $K, N, T, b, c, k = 1$;
while $k < K$ **do**
  2. Calculate α^k using gradient descent in Eq. 2, $n = 1$;
  while $n < N$ **do**
    3. Calculate α_n^k in Eq. 3, $\mathcal{L}_{\text{val}}(w^*(\alpha^k), \alpha^k)$, $\mathcal{L}_{\text{val}}(w^*(\alpha_n^k), \alpha_n^k)$ in Eq. 2;
    if $\mathcal{L}_{val}(w^*(\alpha^k), \alpha^k) > \mathcal{L}_{val}(w^*(\alpha_n^k), \alpha_n^k)$ **then**
      $\alpha^{k+1} = \alpha_n^k$
    else
      4. Calculate p using Eq. 4, and generate a random number r;
      if $r < p$ **then**
        $\alpha^{k+1} = \alpha_n^k$
      else
        $\alpha^{k+1} = \alpha^k$
      end
    end
    5. $T = c \cdot T$, $n = n + 1$;
  end
  6. $k = k + 1$;
end

4. Experiments

In this section, we conduct extensive experiments by applying our SA-NAS optimization algorithm based on the following NAS methods: (1) SA-NAS-a: DARTS [15], (2) SA-NAS-b: P-DARTS [3], (3) SA-NAS-c: PC-DARTS [20], (4) SA-NAS-d: Fair DARTS [4].

4.1. Datasets and Configurations

We use two datasets in the experiments: CIFAR-10 and CIFAR-100. The CIFAR-10 dataset contains 50K training images (a 45K training set and a 5K validation set) and 10K testing images, from 10 classes. The CIFAR-100 dataset contains 50K training images (a 45K training set and a 5K validation set) and 10K testing images, from 100 classes. Our implementation is built on PyTorch [16]. The search space for the candidate operations include: 3×3 and 5×5 separable convolutions, 3×3 and 5×5 dilated separable convolutions, 3×3 max pooling, 3×3 average pooling, and skip connection. During architecture search, the hyper-parameters for the architecture are set in the same way as DARTS, P-DARTS, PC-DARTS, and Fair DARTS. The architecture weights are optimized using SGD with a momentum of 0.9 and a weight decay of 3e-4. The initial learning rate is set to 0.005

with a cosine decay scheduler. The architecture is optimized with the Adam [12] optimizer with a learning rate of 3e-4 and a weight decay of 1e-3. The searched architecture is a stack of 8 cells, with the initial channel number set to 16. The search is performed for $K = 50$ epochs, with a total number of neighbouring architectures of $N = 1$. In Algorithm 1, we set T=1e5, b=1, c=0.98 for searching the best architecture. The batch size is 96 in this stage. During architecture evaluation, we set the initial channel number of the architecture to 36. The searched network is trained from scratch for 600 epochs with a batch size of 96. The experiments are conducted on one Tesla V100 GPU.

4.2. Comparison with State-of-the-art Methods

Table 1 shows the results on CIFAR-10 and CIFAR-100, where we include the classification accuracy (%) on the test set, number of parameters (millions) in the searched architecture, and the search cost in terms of GPU hours.

CIFAR-10. As can be seen, all SA-NAS series models achieve comparable or better performance than the baselines in terms of top-1 accuracy. Typically, our SA-NAS-b with fewer parameters achieves the best top-1 accuracy on the CIFAR-10 dataset while our search cost is 1.4 GPU hours less than P-DARTS [3]. Moreover, our SA-NAS-a outperforms the baseline DARTS [15] by a large margin, *i.e.,* 0.19%. These results demonstrate the advantage of our SA-NAS optimization algorithm over state-of-the-art methods using gradient descent.

CIFAR-100. For the CIFAR-100 dataset, our SA-NAS-a and SA-NAS-b with fewer parameters and less search time also achieve better results in terms of top-1 accuracy compared against baselines. Specifically, our SA-NAS-a outperforms the corresponding baseline, DARTS [15], by 0.21% in terms of top-1 accuracy. After easily adapted to the baselines, our SA-NAS shows its advantage in reducing search time and saving computation resources compared to previous gradient-based methods.

Table 1: Results on CIFAR-10 and CIFAR-100, including classification accuracy (%) on the test set, number of parameters (millions) in the searched architecture, and search cost (GPU hours). Bold and underline denote the first and second place.

Method	N	CIFAR-10			CIFAR-100		
		Top-1(%) ↑	Param.(M) ↓	Cost ↓	Top-1(%) ↑	Param.(M) ↓	Cost ↓
DARTS	0	97.06	3.3	9.6	79.48	<u>1.8</u>	9.6
P-DARTS	0	97.49	3.4	7.2	82.51	3.6	7.2
PC-DARTS	0	97.43	3.6	**2.4**	83.54	3.7	**2.4**
Fair DARTS	0	97.46	<u>3.1</u>	9.6	**83.62**	3.2	9.6
SA-NAS-a	1	97.25	3.2	9.2	79.69	**1.7**	9.2
SA-NAS-b	1	**97.53**	<u>3.1</u>	<u>5.8</u>	82.58	3.1	<u>5.8</u>
SA-NAS-c	1	97.47	3.2	**2.4**	83.56	3.2	**2.4**
SA-NAS-d	1	<u>97.51</u>	**2.7**	8.2	<u>83.61</u>	2.8	8.2

5. Conclusion

In this work, we present a new optimization algorithm based on simulated annealing, called SA-NAS, for neural architecture search to boost search speed and save search computation resources. Specifically, we generate random architectures in the neighborhood of the current best architecture and evaluate their objectives on the validation set. If the objective value is smaller than its current

best value, then the neighboring architecture is accepted; otherwise, the architecture is accepted based on the designed probability density inspired by the Bolzman-Gibbs distribution. Extensive experiments on various benchmarks demonstrate the effectiveness and efficiency of our SA-NAS in achieving comparable or better predictive performance while reducing search costs.

References

[1] Mohamed Abdel-Basset, Weiping Ding, and Doaa El-Shahat. A hybrid harris hawks optimization algorithm with simulated annealing for feature selection. *Artificial Intelligence Review*, pages 593–637, 2021.

[2] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[3] Xin Chen, Lingxi Xie, Jun Wu, and Qi 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)*, pages 1294–1303, 2019.

[4] Xiangxiang Chu, Tianbao Zhou, Bo Zhang, and Jixiang Li. Fair DARTS: eliminating unfair advantages in differentiable architecture search. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 465–480, 2020.

[5] Kathryn A. Dowsland and Jonathan M. Thompson. *Simulated Annealing*, pages 1623–1655. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[6] Simon S Du, Chi Jin, Jason D Lee, Michael I Jordan, Aarti Singh, and Barnabas Poczos. Gradient descent can take exponential time to escape saddle points. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[7] Rong Ge, Furong Huang, Chi Jin, and Yang Yuan. Escaping from saddle points—online stochastic gradient for tensor decomposition. In *Proceedings of the 28th Conference on Learning Theory*, page 797–842, 2015.

[8] Rong Ge, Jason D. Lee, and Tengyu Ma. Matrix completion has no spurious local minimum. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, page 2973–2981, 2016.

[9] Rong Ge, Chi Jin, and Yi Zheng. No spurious local minima in nonconvex low rank problems: A unified geometric analysis. In *Proceedings of the International Conference on Machine Learning (ICML)*, page 1233–1242, 2017.

[10] Xiaoxia Han, Yingchao Dong, Lin Yue, Quanxi Xu, Gang Xie, and Xinying Xu. State-transition simulated annealing algorithm for constrained and unconstrained multi-objective optimization problems. *Applied Intelligence*, 51:775–787, 2021.

[11] Andrew Howard, Ruoming Pang, Hartwig Adam, Mark Sandler, Bo Chen, Weijun Wang, Liang-Chieh Chen, Mingxing Tan, Grace Chu, Vijay Vasudevan, and Yukun Zhu. Searching for MobileNetV3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1314–1324, 2019.

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] Hanwen Liang, Shifeng Zhang, Jiacheng Sun, Xingqiu He, Weiran Huang, Kechen Zhuang, and Zhenguo Li. DARTS+: improved differentiable architecture search with early stopping. *arXiv preprint arXiv:1909.06035*, 2019.

[14] Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical representations for efficient architecture search. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[15] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: differentiable architecture search. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[16] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, pages 8026–8037, 2019.

[17] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the International Conference on Machine Learning (ICML)*, pages 4092–4101. PMLR, 2018.

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

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

[20] 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 *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[21] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

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

Appendix A. Ablation Study

In this part, we perform an ablation study on N, *i.e.* the total number of neighboring architectures and Skip Connection Regularizer (SCR) using SA-NAS-b. We set N =1, 2, 3, 4 and report the mean results on CIFAR-10 and CIFAR-100 in Table 1 using 5 random seeds.

Total number of neighboring architectures. We can observe that with the increase of N, the number of parameters in the searched architecture decreases, but the top-1 accuracy decreases and the search cost increases a lot. This is because more random neighboring architectures deteriorate the best choice of the current best architecture, which demonstrates the importance of having a right choice of N. In order to save computation resources and reduce search cost, we set $N = 1$ in our experiments.

Skip Connection Regularizer. We also explore the importance of the proposed Skip Connection Regularizer (SCR) in our SA-NAS. With the incorporation of SCR, our SA-NAS-b (SCR) achieves slightly better performance than SA-NAS-b while having fewer parameters. This is consistent with our idea to mitigate the negative effect of many skip connections on the oversimplified searched architecture. This also implies the effectiveness of our SA-NAS optimization algorithm in neural architecture search.

Table 2: Ablation Study on N and SCR, where SCR denotes Skip Connection Regularizer.

Method	N	CIFAR-10			CIFAR-100		
		Top-1(%) ↑	Param.(M) ↓	Cost ↓	Top-1(%) ↑	Param.(M) ↓	Cost ↓
SA-NAS-b	1	97.53	3.1	5.8	82.58	3.1	5.8
SA-NAS-b	2	97.52 (↓ 0.01)	2.8 (↓ 0.3)	7.2 (↑ 1.4)	82.61 (↑ 0.03)	3.0 (↓ 0.1)	7.2 (↑ 1.4)
SA-NAS-b	3	97.43 (↓ 0.10)	2.7 (↓ 0.4)	8.5 (↑ 2.7)	82.46 (↓ 0.12)	**2.9** (↓ 0.2)	8.5 (↑ 2.7)
SA-NAS-b	4	97.26 (↓ 0.27)	**2.6** (↓ 0.5)	9.8 (↑ 4.0)	82.39 (↓ 0.19)	**2.9** (↓ 0.2)	9.8 (↑ 4.0)
SA-NAS-b (w. SCR)	1	**97.55** (↑ 0.02)	2.9 (↓ 0.2)	**5.6** (↓ 0.2)	**82.66** (↑ 0.08)	3.0 (↓ 0.1)	**5.6** (↓ 0.2)

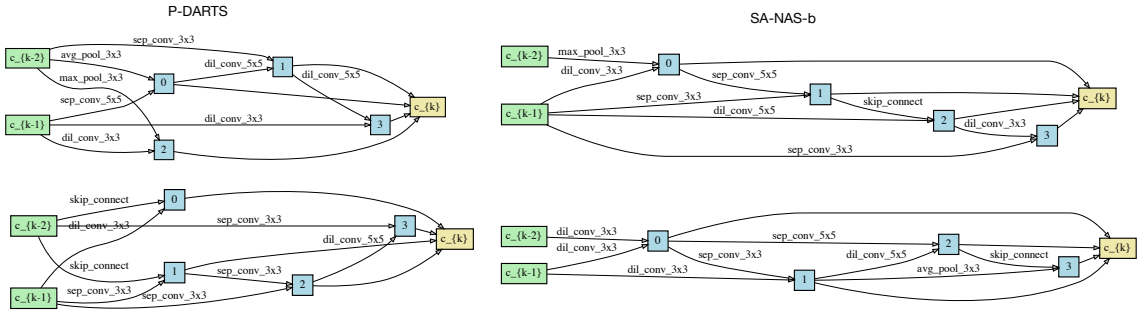

Figure 1: Visualization results of the architecture searched by P-DARTS (Left) and SA-NAS-b (Right). **Top Row**: reduction cell; **Bottom Row**: normal cell.

Appendix B. Visualization

For better comparison, we also visualize the searched architecture using P-DARTS [3] and SA-NAS-b in Figure 1. Specifically, we plot the reduction cell and normal cell searched on the CIFAR-10

benchmark under the same setting for a fair comparison. Note that our SA-NAS-b is based on P-DARTS by applying the proposed SA-NAS optimization algorithm on their gradient descent methods.

As can be seen in Figure 1, our SA-NAS-b achieves fewer skip connection operators in the normal cell, which can avoid the oversimplified solutions of many skip connections for boosting the predictive performance. The depth in our SA-NAS-b is larger than P-DARTS [3] in terms of both reduction and normal cell, which validates the rationality of our SA-NAS in boosting the predictive performance with less search cost and parameters. These visualization results also provide much solid evidence for the comparison results of predictive performance reported in Table 1, where all SA-NAS series models achieve competitive performance than the baselines in terms of top-1 accuracy.