Contemporary Mathematics
https://ojs.wiserpub.com/index.php/CM/

Research Article

SR-DARTS: Robust Differentiable Neural Architecture Search with Stability Regularization

Huakun Wu, Gusheng Tian, Peng Zhang*

Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou, 510006, China
E-mail: p.zhang@gzhu.edu.cn

Received: 25 August 2025; **Revised:** 20 November 2025; **Accepted:** 21 November 2025

Abstract: Neural Architecture Search (NAS) has emerged as a powerful tool to find the best neural architectures with minimal manual intervention. However, existing NAS approaches suffer from poor robustness, because the search process is sensitive to the noise and hyperparameter settings. To this end, we propose a robust Differentiable Architecture Search with Stability Regularization (SR-DARTS for short). SR-DARTS introduces a new stability regularization term into the validation loss function to explicitly constrain the architecture parameter. Specifically, the new regularizer drives the mean of the architecture parameters towards zero during the search process, thereby accelerating the convergence of the search algorithm. Consequently, SR-DARTS can reduce the variance error during search. Experimental results demonstrate that SR-DARTS can discover accurate neural network architectures and obtain better performance on benchmark datasets.

Keywords: neural architecture search, stability regularization

MSC: 37C15, 47A52, 47A64

1. Introduction

Neural Architecture Search (NAS) has attracted considerable attention. Early NAS approaches are mainly based on Reinforcement Learning (RL) [1–3] and Evolutionary Algorithms (EA) [4–6]. Recurrent Neural Network (RNN) controllers are commonly used to create new network architectures, encoded as strings for the agent to interact with, while model evaluation accuracy acts as the reward signal for training the agent. EA-based NAS attempts to jointly optimize network weights and architecture parameters. However, recent studies tend to use EA solely for architecture optimization. Although RL and EA methods provide better accuracy than manually designed architectures, they are highly computationally intensive. For instance, discovering NASNet [7] via RL required up to 2,000 Graphics Processing Unit (GPU)-days. Meanwhile, NAS techniques have also been extended to Graph Neural Networks (GNNs) and heterogeneous graphs [8–11]. These studies further underline the need for robust and transferable architecture search methods on complex graph data.

To alleviate the excessive computational cost and inefficiency of NAS methods based on RL and EA discussed above, Differentiable Architecture Search (DARTS) [12] has been proposed that relaxes the discrete selection of operations into a continuous optimization problem, allowing the architecture parameters to be learned more efficiently. With fewer

DOI: https://doi.org/10.37256/cm.7120268384

GPU computations, DARTS can achieve a performance comparable to the RL and EA methods. However, despite the advantages of differentiable methods in terms of simplicity and computational efficiency, these methods still face challenges in robustness and architecture generalization. Researchers have explored various improvements, such as restricting skip connections and regularizing parameters, but explicit regularization in architecture parameter optimization is rarely involved [13]. Furthermore, DARTS has shown good performance on a single dataset but exhibits poor generalization ability on new datasets. For example, GeNAS [14] achieves excellent performance on Canadian Institute For Advanced Research (CIFAR)-10, but it underperforms on ImageNet. This indicates that learning an architecture with generalizability from a single dataset remains an unresolved challenge.

In order to solve the above limitation, prior works introduce several different regularization methods. For example, fairness reweighting to suppress the dominance of skip connections [15], grouped operation dropout to reduce co-adaptation [16], and constraints on operation strength or spatial uniformity to curb preference [17, 18]. These strategies effectively mitigate certain degenerate behaviors in specific settings, but they are often tailored to particular operators or search spaces and may introduce additional hyperparameters that themselves require careful tuning. Moreover, most of them act indirectly on the search dynamics by modifying operation weights or local preferences, rather than modeling the global distribution of architecture parameters. As a result, the architecture optimization process can still be vulnerable to noisy gradients and biased updates, and may converge to unstable or dataset-specific solutions. However, regularization aimed at the architecture parameters α typically defaults to applying L_2/weight decay to α, rather than imposing an explicit global constraint.

This paper aims to address the aforementioned challenge efficiently. Inspired by commonly utilized L_2 or weight decay regularization techniques, this paper introduces a new differentiable architecture search by adding a Stability Regularization term, called SR-DARTS. In this method, we add a stability regularization term to the DARTS validation loss function to explicitly constrain the architecture parameters, thereby reducing the variance and enhancing the network architecture with better robustness and generalization. The results of the experiments indicate that SR-DARTS surpasses other differentiable search methods by achieving higher accuracy and stronger performance with reduced search time. The main contributions are summarized as follows:

• This paper presents a new neural architecture search algorithm SR-DARTS by introducing a stability regularization to constrain the architecture parameters. SR-DARTS can alleviate the bias of differential search algorithms and improve the robustness of DARTS.

• Experiments validate the effectiveness of the proposed method on real-world datasets. The results in the benchmark datasets show that SR-DARTS not only reduces search time but also improves the accuracy of the model compared to existing DARTS variants.

2. Related works

Differentiable NAS relaxes the discrete selection of operators into a continuous parameterization, where the architecture parameters α define softmax weights over candidate operations. At the level of the computation graph, each edge is instantiated as a mixed operation governed by these weights. The search proceeds as a bilevel optimization: network weights w are updated on the training set, while α is updated by minimizing the validation loss. After convergence, the architecture is discretized by selecting the highest-weight operation on each edge, and the resulting network is retrained from scratch. Compared with reinforcement-learning or evolutionary approaches, this gradient-based framework substantially reduces search cost while maintaining competitive accuracy.

Regularization in DARTS aims to stabilize the dynamics of bilevel optimization. By constraining the architecture parameters α, smoothing the validation-loss landscape, and reducing the unfair advantage of parameter-free operators (e.g., skip connections), it balances competition among candidate operations. Typical mechanisms include penalties on α, fairness reweighting, stochastic perturbations, progressive or discretized search, and constraints on operation strength or uniformity.

Differentiable-NAS Algorithms: Liu et al. [12] pioneered DARTS, which revolutionized the NAS field with a novel differentiable framework. By relaxing the space into a continuous domain and applying gradient-based optimization, DARTS substantially cuts the search expense. However, performance collapse issues in practical applications prompted subsequent improvements. In view of this, Li et al. [19] introduced Progressive Discretization (PD)-DARTS, which employs a progressive search architecture, gradually increasing network depth and restricting skip connections to narrow the gap between the supernet and the search network. Similarly, Liang et al. [20] presented DARTS+, directly limiting the number of skip connections within each cell to prevent performance degradation caused by excessive skip connections. Furthermore, Zela et al. [21] proposed Robustifying (R)-DARTS, which uses Hessian eigenvalues as a collapse indicator and employs stronger regularization to reduce this eigenvalue. Xie et al. [22] developed Stochastic Neural Architecture Search (SNAS), adding perturbations to architecture parameters for implicit regularization to avoid overfitting of architecture parameters. In a related vein, Chu et al. [15] introduced Fair DARTS, which uses independent sigmoid functions to weight operations and prevent unfair competition among different operations. Hong et al. [16] brought forth DropNAS, focusing on co-adaptation problems and the Matthew effect by randomly dropping certain operations during training, enabling the model to learn the importance of different operations more evenly. In summary, these differentiable NAS methods, each with unique advantages in improving search efficiency and model performance, offer diverse and effective strategies for the NAS field.

Regularized Differentiable-NAS Algorithms: Yang et al. [17] proposed Operation Strength (OSTR)-DARTS, which imposes explicit constraints on operation strength to prevent a single operator (e.g., a skip connection) from dominating and to mitigate early bias. Further, Huang et al. [18] proposed Uniform (U)-DARTS, which enforces spatial uniformity/consistency constraints to balance competition among operators and suppress preference at the distribution level. In contrast to the above deterministic-constraint approach, Bi et al. [23] proposed Stabilizing (S)-DARTS, which injects adversarial or stochastic perturbations around the validation objective to smooth curvature and suppress drift of the architecture parameters, thereby reducing the probability of collapse. By comparison, Chen et al. [24] studied Dirichlet Neural Architecture Search (DrNAS), which models the architecture parameters α as Dirichlet random variables with a prior and improves stability and exploration through distributional learning. Meanwhile, Chu et al. [25] proposed Noisy Differentiable Architecture Search (NDAS), which injects unbiased noise into the update of the architecture parameters as an implicit regularizer to further alleviate instability. In summary, these regularization-based differentiable NAS methods enhance the robustness of differentiable NAS through dimensions such as strength constraints and uniformity constraints.

Overall, differentiable NAS relaxes the discrete search space into a continuous one and optimizes it with gradients, typically via bilevel optimization, which updates network weights w on the training set and the architecture parameters α on the validation loss, together with temperature-controlled soft choices (e.g., Gumbel Softmax). This markedly reduces search time while maintaining or improving accuracy. Regularized differentiable NAS further introduces regularization within this framework, aiming to mitigate skip-connection bias, smooth the validation-loss landscape, and balance operator competition. As a result, it preserves the efficiency of differentiable NAS while enhancing robustness and cross-dataset performance.

3. Preliminaries

3.1 *The DARTS method*

First, we introduce the core concepts of the differentiable neural architecture search methods. DARTS [12] tackles the discrete nature of architecture search by relaxing the problem into a continuous space. This allows architecture parameters to be optimized alongside network weights via gradient descent, enhancing the efficiency of finding optimal network architectures.

In the formulation of DARTS, each cell is cast as a directed acyclic graph: a vertex $x^{(i)}$ denotes the feature representation at step i, and a directed link (i, j) applies an operator $o^{(i,j)}$ from $\mathcal{O}$ (e.g., 3×3 convolutions or a skip connection). Two inputs are taken from the preceding cells and a single output is formed by concatenating every intermediate node. The j-th intermediate representation is computed as:

$$x^{(j)} = \sum_{i<j} g^{(i,j)}(x^{(i)}) \tag{1}$$

Here, $g^{(i,j)}(\cdot)$ denotes the mixture of candidate operations on edge (i, j). Let $\alpha^{(i,j)}$ denote the operation weight vector from node $x^{(i)}$ to node $x^{(j)}$, with a dimension of $|\mathcal{O}|$. For simplicity, we denote this as α. The continuous relaxation of the search space is achieved through the softmax operation over the architecture parameters α, which can be specifically expressed with the following formula:

$$g^{(i,j)}(x) = \sum_{o\in\mathcal{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o'\in\mathcal{O}} \exp(\alpha_{o'}^{(i,j)})} o(x) \tag{2}$$

Under this continuous relaxation, both the network weights and the operation-mixing coefficients of the supernet are differentiable. Consequently, differentiable NAS is cast as a bilevel optimization problem:

$$\min_{\alpha} \mathscr{L}_{\text{val}}(w^*(\alpha), \alpha) \tag{3}$$

$$s.t. \quad w^*(\alpha) = \arg\min_{w} \mathscr{L}_{\text{train}}(w, \alpha) \tag{4}$$

where w denotes the supernet weights, $\mathscr{L}_{\text{val}}$ and $\mathscr{L}_{\text{train}}$ are the validation and training loss, respectively. In DARTS, the update for α is computed using the following approximation:

$$\nabla_\alpha \mathscr{L}_{\text{val}}(w', \alpha) - \frac{\xi}{2\varepsilon}\left(\nabla_\alpha \mathscr{L}_{\text{train}}(w^+, \alpha) - \nabla_\alpha \mathscr{L}_{\text{train}}(w^-, \alpha)\right) \tag{5}$$

where ε is a small step size, and $w^{\pm} = w \pm \varepsilon \nabla_{w'} \mathscr{L}_{\text{val}}(w', \alpha)$.

3.2 L_2 regularizer for DARTS

Popular regularizers are the L_2 regularizer and the decay penalty regularizer. Then, the update step of α is denoted as follows:

$$\alpha_k^{t+1} \leftarrow \alpha_k^t - \eta_\alpha \cdot \nabla_{\alpha_k} \mathscr{L}_{\text{val}} \tag{6}$$

where η_α is the learning rate, and $\mathscr{L}_{\text{val}}$ denotes the loss on the validation set.

In the DARTS variant with L_2 regularization, Adam optimizer with L_2 regularization is utilized for the optimization of architecture parameters. The L_2 regularization gradients are normalized by their cumulative magnitude N, which results in relatively uniform penalization for each parameter element. This, in turn, to some extent weakens the effectiveness of regularization [26]. The update formula is defined as follows:

$$\alpha_k^{t+1} \leftarrow \alpha_k^t - \eta_\alpha \cdot \nabla_{\alpha_k} \mathscr{L}_{\text{val}} - \eta_\alpha \lambda N(\alpha_k^t) \tag{7}$$

where η_α denotes the learning rate for the architecture parameters, λ represents the regularization strength, and $\mathscr{L}_{\text{val}}$ is the validation loss function.

In the weight decay regularization method, a portion of the architecture parameters α is directly subtracted during the optimization steps. At each iteration, we first compute the gradient of the validation loss with respect to α and update it along the negative gradient direction, and then subtract an additional term proportional to the current parameter value and the weight decay coefficient. This causes the parameters to gradually decay during updates, thus achieving the effect of L_2 regularization. The update formula is as follows:

$$\alpha_k^{t+1} \leftarrow \alpha_k^t - \eta_\alpha \cdot \nabla_\alpha \mathscr{L}_{\text{val}} - \eta_\alpha \cdot \tau \cdot \alpha_k^t \tag{8}$$

where η_α denotes the learning rate for the architecture parameters, τ represents the weight decay coefficient, and $\mathscr{L}_{\text{val}}$ is the validation loss function.

4. The proposed method

Although DARTS has achieved significant progress in architecture search, some challenges remain in practical applications. For example, the uncontrolled growth of architecture parameters α often results in an over-preference to simple operations (e.g., skip connections). To address this issue, we introduce a stability regularization term into the validation loss function of DARTS by constraining the mean value of architecture parameters, guiding it toward a target mean μ. Consequently, the validation loss takes the form:

$$\mathscr{L}_{\text{val}}(\alpha) = \mathscr{L}_{\text{val}}(w^*(\alpha), \alpha) + \lambda \left(\mu - \text{mean}(\alpha)\right)^2 \tag{9}$$

Here, μ is the target mean set to 0, λ is the hyperparameter controlling the regularization strength, and $\text{mean}(\alpha)$ represents the mean value of the current architecture parameters, calculated as:

$$\text{mean}(\alpha) = \frac{1}{|\alpha|} \sum_{k=1}^{|\alpha|} \alpha_k \tag{10}$$

Here, $|\alpha|$ represents the cardinality of the operator set, and α_k denotes the k^{th} component of the architecture vector α. Building on the concept from formula (8), we can expand formula (9) to derive a more concise update formula for α, as shown below:

$$\nabla_\alpha \mathscr{L}_{\text{val}}(\alpha) = \left(\nabla_\alpha \mathscr{L}_{\text{val}}(w', \alpha) - \frac{\xi}{2\varepsilon}\left(\nabla_\alpha \mathscr{L}_{\text{train}}(w^+, \alpha) - \nabla_\alpha \mathscr{L}_{\text{train}}(w^-, \alpha)\right)\right) - \lambda \cdot \frac{2}{|\alpha|}\left(\mu - \frac{1}{|\alpha|}\sum_{k=1}^{|\alpha|} \alpha_k\right) \tag{11}$$

where w^+ and w^- denote the weights obtained after applying a positive and a negative perturbation to the architecture parameters α, respectively. ξ and ε are hyperparameters that control the update step size.

Algorithm 1 summarizes the overall procedure of SR-DARTS. Given the architecture coefficient vector α, supernet weights w, and the number of search epochs E, we first instantiate the supernet and initialize α and w. At each epoch, α is updated by gradient descent on the validation objective augmented with the stability term $\lambda\left(\mu - \text{mean}(\alpha)\right)^2$ (with $\mu = 0$ by default), followed by a standard update of w on the training loss. After E epochs, α is discretized to obtain the

final architecture. This alternating scheme is used in the bilevel optimization while keeping the overall search pipeline identical to DARTS. Regarding time complexity, T_{val} and T_{train} denote the time cost of one forward-backward pass of the supernet on the validation and training sets, respectively. During the search process, in each epoch Algorithm 1 performs one update of the architecture parameters α (cost T_{val}) and one update of the supernet weights w (cost T_{train}), and hence the overall time complexity is $\mathcal{O}\big(E \cdot (T_{\text{val}} + T_{\text{train}})\big)$. The additional stability regularization term only introduces an extra $\mathcal{O}(|\alpha|)$ operation in the α-update, and therefore the time complexity remains of the same order as vanilla DARTS.

We now use an example to explain the main steps of SR-DARTS algorithm. Consider the CIFAR-10 [17] dataset, which contains 10 classes of 32 × 32 Red Green Blue (RGB) images. The search space adopts a DARTS-style cell and includes eight candidate operations: 3 × 3 and 5 × 5 standard convolutions, dilated convolutions, depthwise-separable convolutions, 3 × 3 max/average pooling, the identity (skip) operation, and the zero operation (each followed by Batch Normalization (BN) and Rectified Linear Unit (ReLU)). Table 1 provides the detailed specifications of these operators.

Algorithm 1 Stability-Regularized Differentiable Architecture Search (SR-DARTS)

Require: Architecture parameters α, Supernet weights w, Total search epochs E

1: Construct the supernet and initialize (α, w).

2: **for** each $e = 1, \ldots, E$ **do**

3: Update α by performing gradient descent on $\nabla_\alpha(\mathcal{L}_{\text{val}} + \lambda(\mu - \text{mean}(\alpha))^2)$

4: Update w by performing gradient descent on $\nabla_w \mathcal{L}_{\text{train}}$

5: **end for**

6: Decode the architecture parameters α to a discrete design and output the final architecture.

Table 1. Convolutional Neural Network (CNN) architecture search space (DARTS-style)

Component	Values
Input nodes	Two inputs c_{k-1} and c_{k-2} (outputs of the $(k-1)^{th}$ and $(k-2)^{th}$).
Operation set (per edge)	Sep_conv_3 × 3, sep_conv_5 × 5, dil_conv_3 × 3, dil_conv_5 × 5, max_pool_3 × 3, avg_pool_3 × 3, skip_connection (identity), zero.
Cell topology	Four intermediate nodes; each node selects two inputs; output is concat of all nodes.
Reduction cell	Stride-two counterparts of convolution/pooling operations.
Post-ops	Each operation followed by BN and ReLU.

Table 2. SR-DARTS discovered cells on CIFAR-10 in the first epoch

Node	Normal cell		Reduction cell	
	Input 1 (source, op)	Input 2 (source, op)	Input 1 (source, op)	Input 2 (source, op)
0	$(c_{k-2}$, sep_conv_5 × 5)	$(c_{k-1}$, sep_conv_5 × 5)	$(c_{k-1}$, sep_conv_5 × 5)	$(c_{k-2}$, sep_conv_3 × 3)
1	(0, sep_conv_5 × 5)	$(c_{k-1}$, dil_conv_3 × 3)	(0, sep_conv_5 × 5)	$(c_{k-2}$, sep_conv_3 × 3)
2	(1, dil_conv_5 × 5)	(identity)	(1, dil_conv_5 × 5)	(0, dil_conv_3 × 3)
3	(identity)	(identity)	(0, dil_conv_5 × 5)	(1, dil_conv_3 × 3)

With the search space in Table 1 fixed, SR-DARTS performs a bilevel optimization: at each epoch the architecture parameters α are updated on the validation objective augmented with the stability term $\lambda(\mu - \text{mean}(\alpha))^2$, while the network weights w are updated on the training loss. To visualize the early search behavior, after the first epoch we temporarily discretize the supernet following the DARTS rule: for each intermediate node we retain the two incoming

edges with the largest non-zero softmax weights and label each kept edge with the operation attaining the highest weight (the reduction cell uses the stride-two counterparts). The resulting epoch-1 snapshots of the normal and reduction cells are summarized in Table 2. After completing E epochs, we discretize the final architecture and retrain it from scratch, and then report results on CIFAR-10/100.

5. Experimental setup and results

This section presents the detailed information of our experiments. The experiments aim to discover convolutional network architectures for image classification, using the widely adopted CIFAR-10 [17] and CIFAR-100 [18] benchmarks. In Section 5.2, we search for and evaluate architectures on the CIFAR-10 and CIFAR-100 datasets. Then, to verify the generalization ability of the architectures discovered by our method, architectures found on CIFAR-10 are applied to CIFAR-100, and vice versa. Finally, the ablation study is presented in Section 5.3.

5.1 *Experimental setup*

We study neural architecture search on the CIFAR-10/100 benchmarks. Both corpora comprise 32×32 RGB images; CIFAR-10 contains 60 K examples over 10 classes (≈ 6 K per class, split 5 K/1 K for train/test), and CIFAR-100 spans 100 classes with ≈ 600 images per class (500/100). The candidate cell follows a DARTS-style space with eight operators: 3×3 and 5×5 convolutions in standard, dilated, and depthwise-separable forms; 3×3 max and average pooling; an identity (skip) connection; and a zero operation. Batch normalization and ReLU are applied after each operation [27, 28]. During the search stage, we train a network with a 16-channel stem for 50 epochs and optimize the architecture parameters α using Adam (learning rate 6×10^{-4}, $\beta_1 = 0.5$, $\beta_2 = 0.999$, weight decay 10^{-3}) with mini-batches of 256. Following the common DARTS protocol, the search is carried out on CIFAR-10; the discovered cell is then instantiated as a full network and trained from scratch, and results are reported on both CIFAR-10 and CIFAR-100.

5.2 *Comparative study*

As shown in Table 3, we benchmark our approach against representative NAS algorithms. We conduct searches on CIFAR-10 and CIFAR-100 and evaluate the obtained architectures on both. Our proposed algorithm achieves a test accuracy of 97.30% on CIFAR-10, showing a slight improvement over DARTS. In CIFAR-100, it also achieves a test accuracy of 83.05%, showing a significant improvement over DARTS. The search cost of SR-DARTS is reduced to 0.5 GPU-days, notably less than the 4 GPU-days required by DARTS. Compared to the RL-DARTS algorithm, our algorithm achieves better search results in CIFAR-10 and CIFAR-100 with a comparable search cost. Figure 1 and Figure 2 present the normal cell and reduction cell searched by SR-DARTS on CIFAR-10 and CIFAR-100, respectively. Additionally, Figures 3-6 include snapshots of the normal and reduction cells on CIFAR-10 and CIFAR-100 after 15, 30, and 45 search epochs, illustrating the evolution of the discovered architectures during search. In terms of generalization ability, the architecture found in CIFAR-100 can still achieve an accuracy of 96.89% on CIFAR-10, and the model found in CIFAR-10 can reach an accuracy of 81.90% on CIFAR-100. SR-DARTS demonstrates excellent performance and strong generalization ability across different datasets. This is primarily attributed to its stability regularization term, which effectively constrains the mean of architecture parameters, thereby reducing fluctuations and deviations.

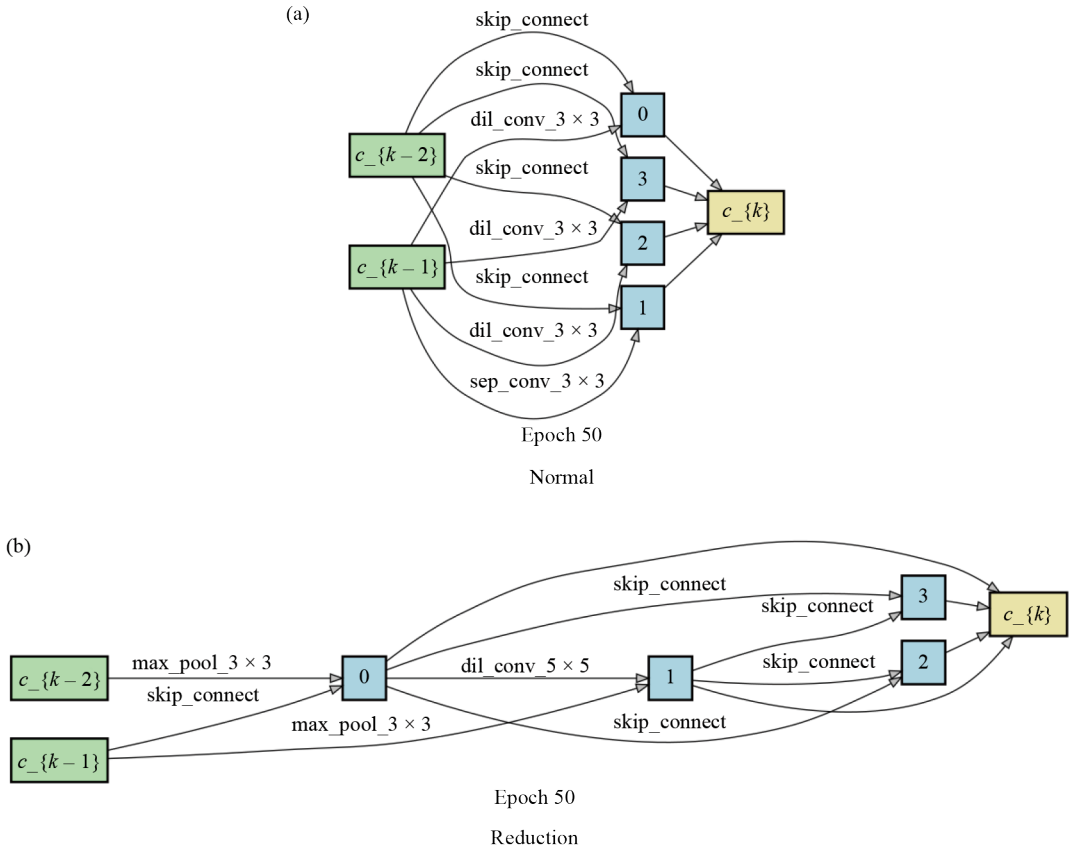


Figure 1. Cells found on CIFAR-10 by SR-DARTS

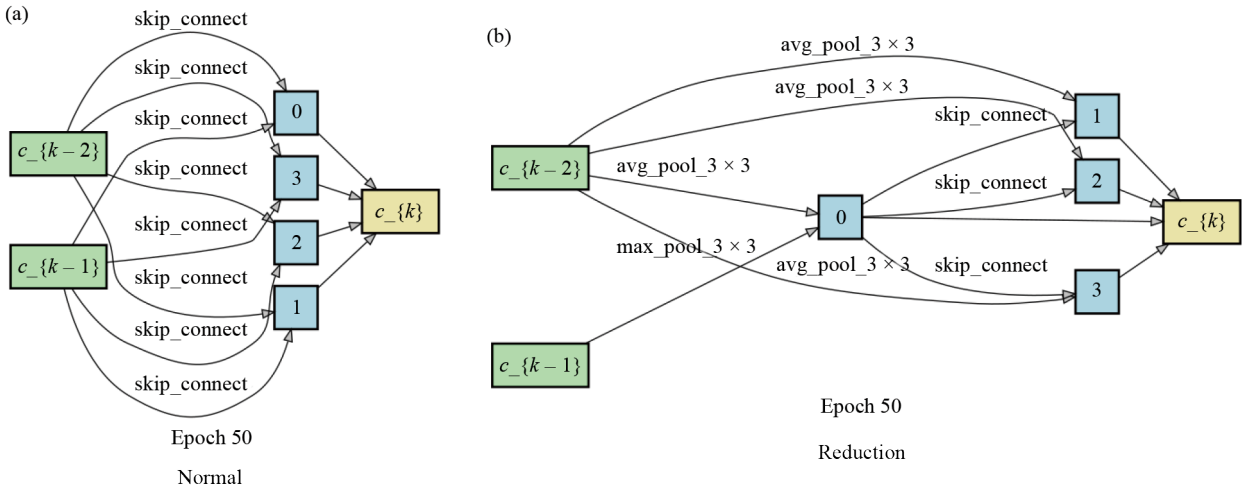


Figure 2. Cells found on CIFAR-100 by SR-DARTS

Table 3. Performance comparison of state-of-the-art architectures on CIFAR-10 and CIFAR-100

Architecture	Test accuracy (%)		Params (*M*)	Search cost (GPU-days)	Search method
	CIFAR-10	CIFAR-100			
ResNet-v2 [29]	94.54	75.67	1.7	-	Manual-designed
Wide Residual Networks (WRN) [30]	95.83	79.50	36.5	-	Manual-designed
DenseNet [29]	96.54	82.82	25.6	-	Manual-designed
ResNeXt [31]	96.42	82.69	68.1	-	Manual-designed
PyramidNet [32]	96.52	82.99	27.0	-	Manual-designed
Broad Neural Architecture Search (BNAS) [33]	97.12	-	4.8	0.19	RL
BNAS-v2 [34]	97.23	-	3.5	0.09	RL
AmoebaNet-A [26]	96.66	-	3.2	3150	Evolution
Progressive Neural Architecture Search (PNAS) [35]	96.59	-	3.2	225	Evolution
AmoebaNet-B [26]	97.45	-	3.2	3150	Evolution
DARTS [12]	97.24	82.52	3.3	4	Gradient-based
R-DARTS [21]	97.05	81.99	-	1.6	Gradient-based
SNAS [22]	97.17	79.91	2.9	1.5	Gradient-based
RL-DARTS [13]	97.26	82.09	3.2	0.18	Gradient-based
SR-DARTS (ours)	97.30	83.05	3.3	0.5	Gradient-based

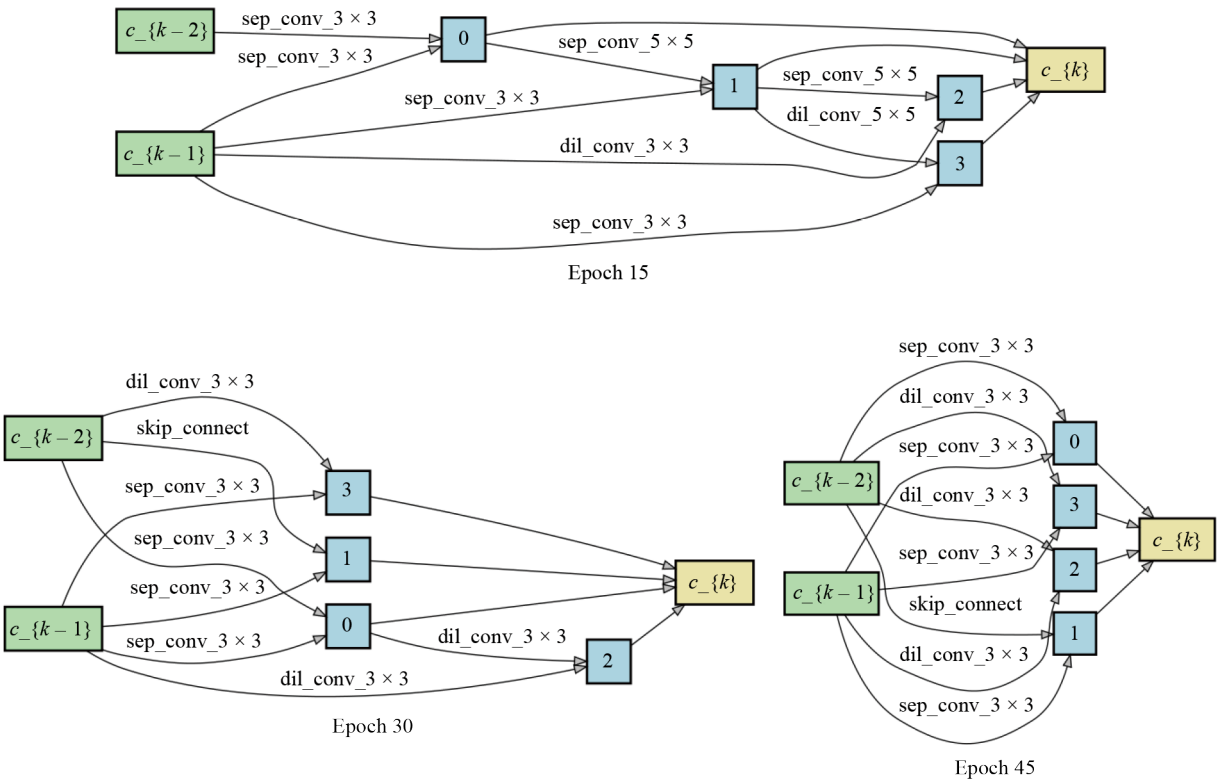
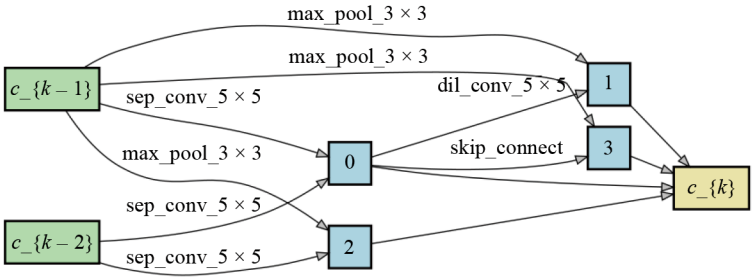
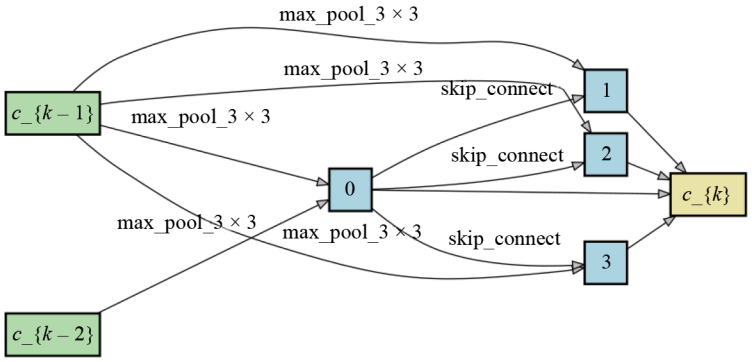


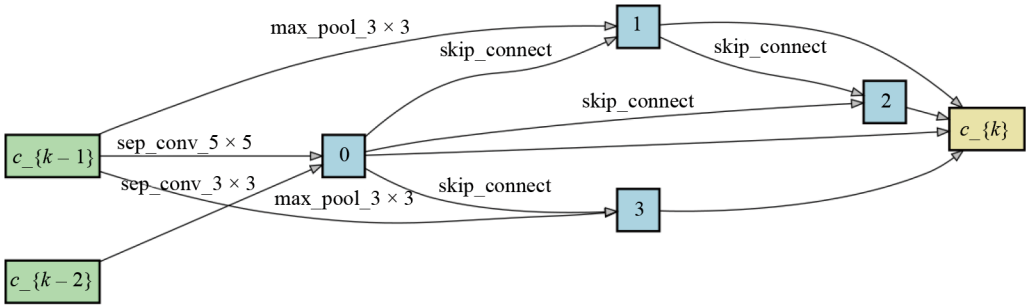
Figure 3. Normal cells found by SR-DARTS on CIFAR-10 during the iterative search



Epoch 15

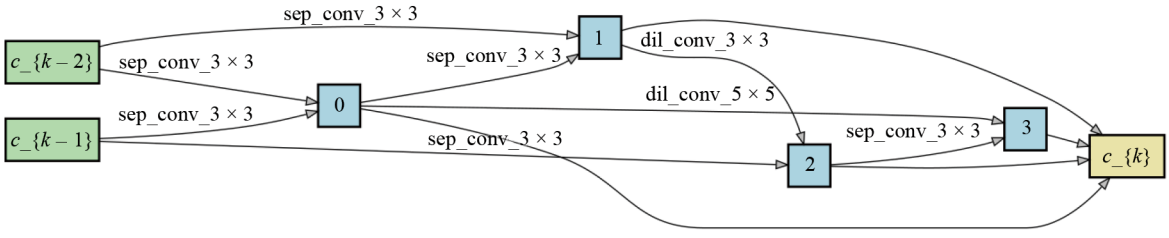


Epoch 30



Epoch 45

Figure 4. Reduce cells found by SR-DARTS on CIFAR-10 during the iterative search



Epoch 15

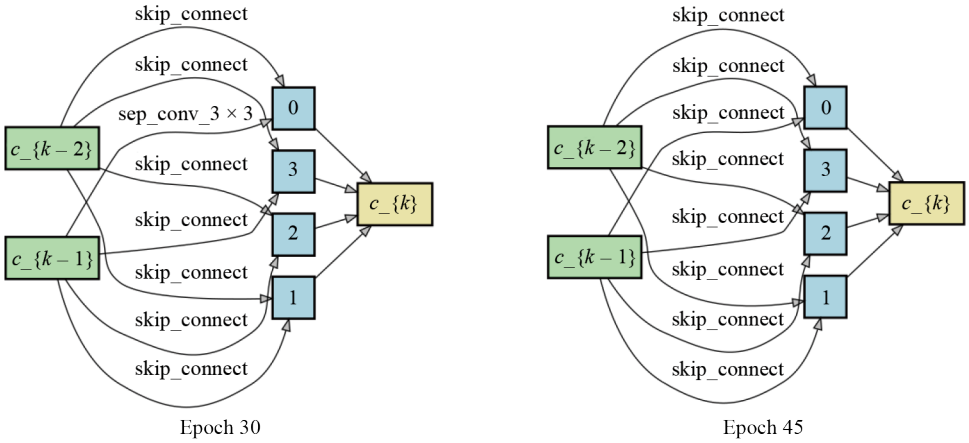


Figure 5. Normal cells found by SR-DARTS on CIFAR-100 during the iterative search

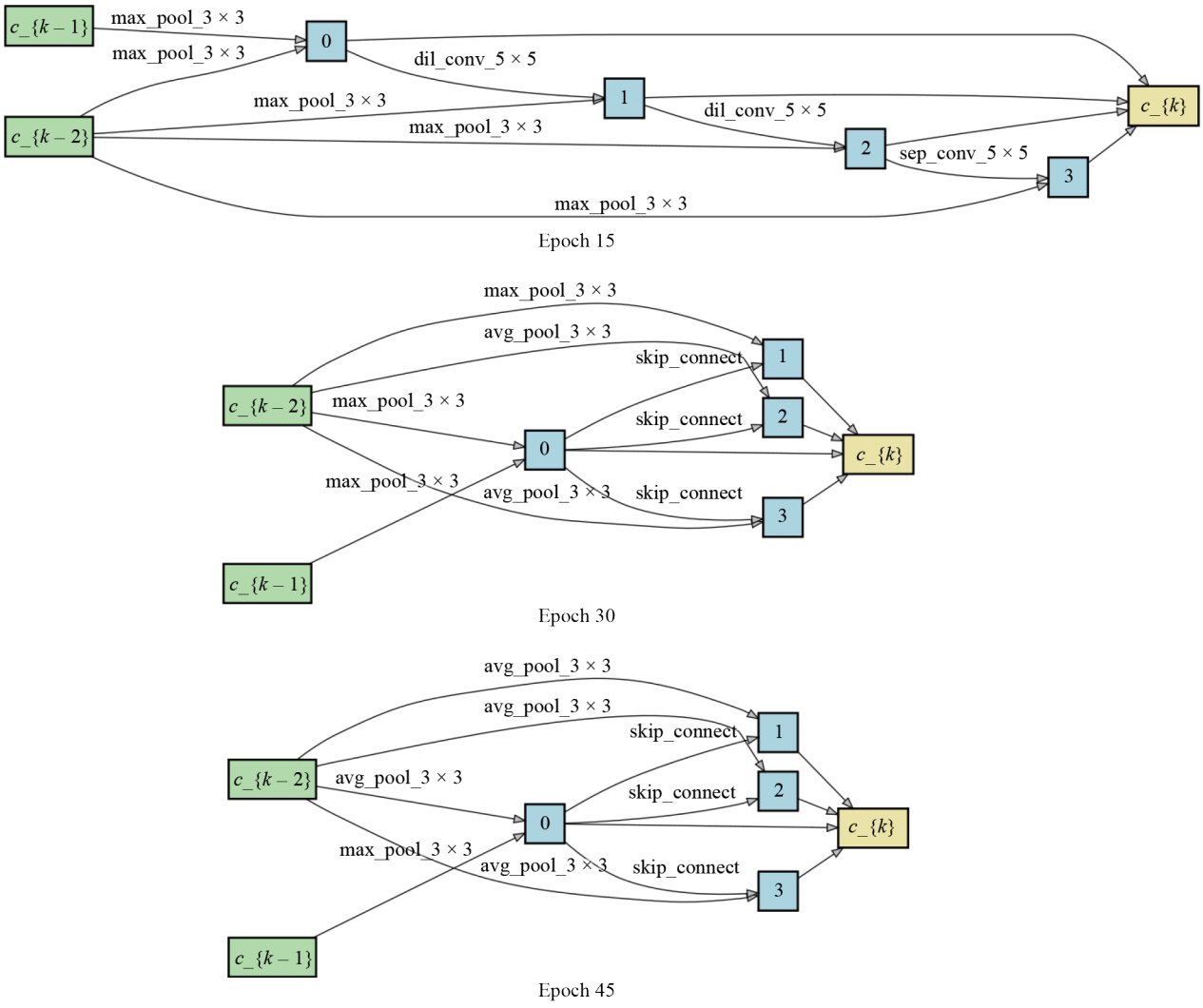


Figure 6. Reduce cells found by SR-DARTS on CIFAR-100 during the iterative search

5.3 *Ablation study*

Table 4 shows the regularization parameter λ of the stability regularization term added to the validation loss function. It can be observed that on both CIFAR-10 and CIFAR-100, the test accuracy increases with the regularization parameter λ and reaches its peak when λ is 0.1. Nevertheless, if the regularization parameter λ is either too large or too small, it may excessively constrain the variation of the architecture parameter, thereby affecting the model's flexibility and leading to a decline in performance. Consequently, identifying an appropriate regularization strength in experiments is of great significance for balancing the model's stability and adaptability.

Table 4. Influence of different regularization parameter schemes on SR-DARTS

Regularization parameter (λ)	0.025	0.050	0.075	0.100	0.125
CIFAR-10 valid (%)	97.18	97.20	97.23	97.30	97.25
CIFAR-100 valid (%)	82.87	82.90	82.99	83.05	83.01

6. Conclusion

In this paper, we propose a robust differentiable neural architecture search algorithm based on stability regularization, named SR-DARTS. By incorporating a stability regularization term into the validation loss function, SR-DARTS explicitly constrains the mean of architecture parameters. This approach reduces their fluctuations or deviations, thereby enhancing the generalization ability of the architecture. Our experimental results demonstrate that SR-DARTS can effectively identify high-performing network architectures and achieve lower error rates on the CIFAR dataset. Future work will focus on optimizing the stability regularization term through dynamic adjustment of the regularization strength and exploring additional regularization strategies to further enhance robustness and generalization.

Acknowledgement

This work was supported in part by Joint Foundation of Guangzhou and Universities on Basic Research (2025A03J-3178).

Conflict of interest

The authors declare no competing financial interest.

References

[1] Ren P, Xiao Y, Chang X, Huang P, Li Z, Chen X, et al. A comprehensive survey of neural architecture search: challenges and solutions. *ACM Computing Surveys*. 2021; 54(4): 34. Available from: https://doi.org/10.1145/3447582.

[2] Wu R, Li C, Zou J, Wang S. Generalizable learning reconstruction for accelerating mr imaging via federated neural architecture search. *arXiv:2308.13995*. 2023. Available from: https://doi.org/10.48550/arXiv.2308.13995.

[3] Chen Y, Yang T, Zhang X, Meng G, Xiao X, Sun J. DETNAS: backbone search for object detection. *arXiv:1903.10979*. 2019. Available from: https://doi.org/10.48550/arXiv.1903.10979.

[4] Liu Y, Sun Y, Xue B, Zhang M, Yen GG, Tan KC. A survey on evolutionary neural architecture search. *IEEE Transactions on Neural Networks and Learning Systems*. 2023; 34: 550-570. Available from: https://doi.org/10.1109/TNNLS.2021.3100554.

[5] Wang B, Pei W, Xue B, Zhang M. Explaining deep convolutional neural networks for image classification by evolving local interpretable model-agnostic explanations. *IEEE Transactions on Emerging Topics in Computational Intelligence*. 2025; 9(6): 3806-3820. Available from: https://doi.org/10.1109/TETCI.2025.3558419.

[6] Molina D, Poyatos J, Del Ser J, García S, Ishibuchi H, Triguero I, et al. Evolutionary computation for the design and enrichment of general-purpose artificial intelligence systems: survey and prospects. *IEEE Transactions on Evolutionary Computation*. 2025. Available from: https://doi.org/10.1109/TEVC.2025.3530096.

[7] Zoph B, Vasudevan V, Shlens J, Le QV. Learning transferable architectures for scalable image recognition. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT, USA: IEEE; 2018. p.8697-8710.

[8] Gao Y, Yang H, Zhang P, Zhou C, Hu Y. GraphNAS: graph neural architecture search with reinforcement learning. *arXiv:1904.09981*. 2019. Available from: https://arxiv.org/abs/1904.09981.

[9] Gao Y, Zhang P, Yang H, Zhou C, Hu Y, Tian Z, et al. GraphNAS++: distributed architecture search for graph neural networks. *IEEE Transactions on Knowledge and Data Engineering*. 2023; 35(7): 6973-6987. Available from: https://doi.org/10.1109/TKDE.2022.3178153.

[10] Gao Y, Zhang P, Zhou C, Yang H, Li Z, Hu Y, et al. HGNAS++: efficient architecture search for heterogeneous graph neural networks. *IEEE Transactions on Knowledge and Data Engineering*. 2023; 35(9): 9448-9461. Available from: https://doi.org/10.1109/TKDE.2023.3239842.

[11] Ji Z, Kong D, Yang Y, Liu J, Li Z. ASSL-HGAT: active semi-supervised learning empowered heterogeneous graph attention network. *Knowledge-Based Systems*. 2024; 290: 111567. Available from: https://doi.org/10.1016/j.knosys.2024.111567.

[12] Liu H, Simonyan K, Yang Y. DARTS: differentiable architecture search. *arXiv:1806.09055*. 2019. Available from: https://doi.org/10.48550/arXiv.1806.09055.

[13] Pang D, Le X, Guan X. RL-DARTS: differentiable neural architecture search via reinforcement-learning-based meta-optimizer. *Knowledge-Based Systems*. 2021; 234: 107585. Available from: https://doi.org/10.1016/j.knosys.2021.107585.

[14] Jeong J, Yu J, Park G, Han D, Yoo Y. GeNAS: neural architecture search with better generalization. *arXiv:2305.08611*. 2023. Available from: https://doi.org/10.48550/arXiv.2305.08611.

[15] Chu X, Zhou T, Zhang B, Li J. Fair DARTS: eliminating unfair advantages in differentiable architecture search. In: *Computer Vision—ECCV 2020*. Springer; 2020. p.465-480. Available from: https://doi.org/10.1007/978-3-030-58555-6_28.

[16] Hong W, Li G, Zhang W, Tang R, Wang Y, Li Z, et al. DropNAS: grouped operation dropout for differentiable architecture search. *arXiv:2201.11679*. 2020. Available from: https://doi.org/10.48550/arXiv.2201.11679.

[17] Yang L, Zheng Z, Han Y, Song S, Huang G, Li F. OStr-DARTS: differentiable neural architecture search based on operation strength. *IEEE Transactions on Cybernetics*. 2024; 54(11): 6559-6572. Available from: https://doi.org/10.1109/TCYB.2024.3455760.

[18] Huang L, Sun S, Zeng J, Wang W, Pang W, Wang K. U-DARTS: uniform-space differentiable architecture search. *Information Sciences*. 2023; 628: 339-349. Available from: https://doi.org/10.1016/j.ins.2023.01.129.

[19] Li Y, Zhou Y, Wang Y, Tang Z. PD-DARTS: progressive discretization differentiable architecture search. In: *Pattern Recognition and Artificial Intelligence*. Springer; 2020. p.306-311. Available from: https://doi.org/10.1007/978-3-030-59830-3_26.

[20] Liang H, Zhang S, Sun J, He X, Huang W, Zhuang K, et al. DARTS+: improved differentiable architecture search with early stopping. *arXiv:1909.06035*. 2020. Available from: https://doi.org/10.48550/arXiv.1909.06035.

[21] Zela A, Elsken T, Saikia T, Marrakchi Y, Brox T, Hutter F. Understanding and robustifying differentiable architecture search. In: *International Conference on Learning Representations*. Addis Ababa, Ethiopia: ICLR; 2020. p.1-28.

[22] Xie S, Zheng H, Liu C, Lin L. SNAS: stochastic neural architecture search. *arXiv:1812.09926*. 2020. Available from: https://doi.org/10.48550/arXiv.1812.09926.

[23] Bi K, Hu C, Xie L, Chen X, Wei L, Tian Q. Stabilizing darts with amended gradient estimation on architectural parameters. *arXiv:1910.11831*. 2020. Available from: https://doi.org/10.48550/arXiv.1910.11831.

[24] Chen X, Wang R, Cheng M, Tang X, Hsieh C-J. DrNAS: dirichlet neural architecture search. In: *International Conference on Learning Representations*. ICLR; 2021. p.1-17.

[25] Chu X, Zhang B. Noisy differentiable architecture search. *arXiv:2005.03566*. 2021. Available from: https://doi.org/10.48550/arXiv.2005.03566.

[26] Real E, Aggarwal A, Huang Y, Le QV. Regularized evolution for image classifier architecture search. *Proceedings of the AAAI Conference on Artificial Intelligence*. 2019; 33(1): 4780-4789. Available from: https://doi.org/10.1609/aaai.v33i01.33014780.

[27] Yu J, Gao Y, Yang H, Tian Z, Zhang P, Zhu X. Automated graph anomaly detection with large language models. *Knowledge-Based Systems*. 2025; 324: 113809. Available from: https://doi.org/10.1016/j.knosys.2025.113809.

[28] Tan W, Yang H, Ye S, Zhang J, Zhang P, Qiao Z. Text-graph alignment based on llms instruction learning and wavelet graph transformers for brain disease analysis. *Neurocomputing*. 2025; 131355. Available from: https://doi.org/10.1016/j.neucom.2025.131355.

[29] Huang G, Liu Z, Van der Maaten L, Weinberger KQ. Densely connected convolutional networks. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition*. Honolulu, HI, USA: IEEE; 2017. p.2261-2269. Available from: https://doi.org/10.1109/CVPR.2017.243.

[30] Zagoruyko S, Komodakis N. Wide residual networks. In: *Proceedings of the British Machine Vision Conference*. York, UK: BMVA Press; 2016. Available from: https://doi.org/10.5244/C.30.87.

[31] Xie S, Girshick R, Dollár P, Tu Z, He K. Aggregated residual transformations for deep neural networks. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition*. Honolulu, HI, USA: IEEE; 2017. p.5987-5995. Available from: https://doi.org/10.1109/CVPR.2017.634.

[32] Han D, Kim J, Kim J. Deep pyramidal residual networks. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition*. Honolulu, HI, USA: IEEE; 2017. p.6307-6315. Available from: https://doi.org/10.1109/CVPR.2017.668.

[33] Ding Z, Chen Y, Li N, Zhao D, Sun Z, Chen CLP. BNAS: efficient neural architecture search using broad scalable architecture. *IEEE Transactions on Neural Networks and Learning Systems*. 2022; 33(9): 5004-5018. Available from: https://doi.org/10.1109/TNNLS.2021.3067028.

[34] Ding Z, Chen Y, Li N, Zhao D. BNAS-v2: memory-efficient and performance-collapse-prevented broad neural architecture search. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*. 2022; 52(10): 6259-6272. Available from: https://doi.org/10.1109/TSMC.2022.3143201.

[35] Liu C, Zoph B, Neumann M, Shlens J, Hua W, Li L-J, et al. Progressive neural architecture search. *arXiv:1712.00559*. 2018. Available from: https://doi.org/10.48550/arXiv.1712.00559.