Neural Networks 158 (2023) 111–120

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

An architecture entropy regularizer for differentiable neural architecture search

Kun Jing, Luoyu Chen, Jungang Xu *

University of Chinese Academy of Sciences, Huaibei Town, Huairou District, Beijing, 101408, China

ARTICLE INFO

Article history:
Received 13 September 2021
Received in revised form 29 August 2022
Accepted 9 November 2022
Available online 16 November 2022

Keywords:
Differentiable architecture search
Matthew effect
Discretization discrepancy
Architecture entropy regularizer

ABSTRACT

Differentiable architecture search (DARTS) is one of the prevailing paradigms of neural architecture search (NAS) due to allowing efficient gradient-based optimization during the search phase. However, its poor stability and generalizability are intolerable. We argue that the crux is the locally optimal architecture parameter caused by a dilemma, which is that the solutions to the Matthew effect and discretization discrepancy are inconsistent. To escape from the dilemma, we propose an architecture entropy to measure the discrepancy of the architecture parameters of different candidate operations and use it as a regularizer to control the learning of architecture parameters. Extensive experiments show that an architecture entropy regularizer with a negative or positive coefficient can effectively solve one side of the contradiction respectively, and the regularizer with a variable coefficient can relieve DARTS from the dilemma. Experimental results demonstrate that our architecture entropy regularizer can significantly improve different differentiable NAS algorithms on different datasets and different search spaces. Furthermore, we also achieve more accurate and more robust results on CIFAR-10 and ImageNet. The code is publicly available at https://github.com/kunjing96/DARTS-AER.

1. Introduction

Neural architecture search (NAS) has emerged as a prominent approach to automatic architecture design, which takes an important step in deep learning. Early NAS algorithms search straightforwardly in the discrete architecture space through reinforcement learning (Zoph & Le, 2017; Zoph, Vasudevan, Shlens, & Le, 2018) and evolutionary algorithm (Liu, Simonyan, Vinyals, Fernando, & Kavukcuoglu, 2018; Real et al., 2017). These algorithms require massive computing resources because training candidate architectures from scratch is a computing resource-intensive task. The introduction of the weight-sharing technique (Pham, Guan, Zoph, Le, & Dean, 2018) reduces the search cost. Based on it, DARTS (Liu, Simonyan, & Yang, 2019) further builds a continuous mixture of architectures and relaxes the categorical architecture search problem to learn differentiable architecture parameters.

Although DARTS has high computational efficiency, it has been criticized for its poor stability and generalizability. Some works (Chen, Xie, Wu, & Tian, 2019; Chu, Zhou, Zhang, & Li, 2020; Hong et al., 2020; Li, Zhang, Wang, Li, & Zhang, 2019; Liang et al., 2019; Zheng et al., 2020) found two problems leading to the poor stability and generalization of DARTS: the rich-get-richer and the large gap between search and evaluation scenarios (described in Section 3). Meanwhile, many improvements are proposed and proved to be effective, including gradually pruning (Chen et al., 2019; Li et al., 2019; Zheng et al., 2020), pre-training before search (Chen et al., 2019), candidate operation grouping (Hong et al., 2020; Li et al., 2019), early stopping (Chu, Zhou et al., 2020; Liang et al., 2019), and restricting the number of skip connections (Chen et al., 2019; Chu, Zhou et al., 2020; Liang et al., 2019). However, the locally optimal architecture parameters is the crux of the problems, which is not solved in these works. The reasons may include: (1) they do not realize that these solutions to the two problems are inconsistent, i.e., the requirements of fair training (solution to problem 1) and high self-confidence selection (solution to problem 2) for architecture parameters are contradictory, where the former requires the entropy of architecture parameters to be as large as possible and the latter expects the entropy of architecture parameters to be as small as possible; (2) they do not directly improve the learning of architecture parameters but indirectly remedy the learning process.

In this work, we redefine these two problems as the Matthew effect and discretization discrepancy and argue that the failure of architecture parameters learning is the fundamental cause of these two problems. To control the learning of architecture parameters, we define the average of the entropy of the normalized architecture parameters for all edges in the cell as the architecture entropy and propose an architecture entropy regularizer with a negative or positive coefficient, which can alleviate the Matthew effect (negative) and the discretization discrepancy

* Corresponding author.
E-mail addresses: jingkun18@mails.ucas.ac.cn (K. Jing), chenluoyu19@mails.ucas.ac.cn (L. Chen), xujg@ucas.ac.cn (J. Xu).

https://doi.org/10.1016/j.neunet.2022.11.015
0893-6080/

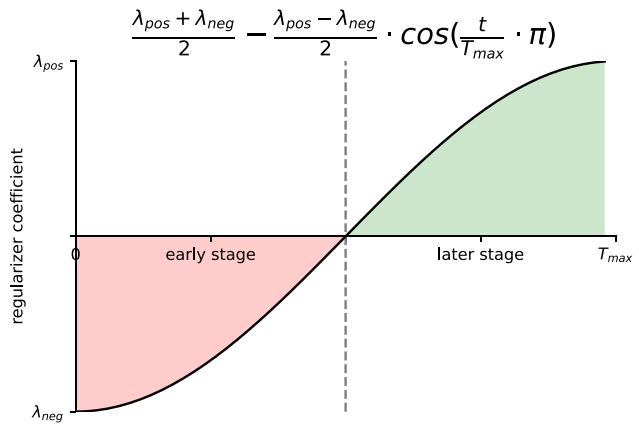


Fig. 1. The scheduling of architecture entropy regularizer coefficient during the search phase. The red area (the left of dashed line) indicates that in the early stage, the regularizer coefficient increases from λ_{neg} to 0, gradually slowing down the restrictions on the dominant expression of candidate operation. The green area (the right of dashed line) indicates that in the later stage, the regularizer coefficient increases from 0 to λ_{pos}, gradually enhancing the rewards on the dominant expression of candidate operation. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

(positive) respectively. Through observing their effects on searching at different stages, we achieve a trade-off in the dilemma by scheduling the regularizer coefficient from negative to positive during the search phase, as shown in Fig. 1.

Extensive experiments prove that an architecture entropy regularizer with a negative or positive coefficient can effectively solve the Matthew effect and discretization discrepancy respectively. Meanwhile, the regularizer with a variable coefficient relieves DARTS from a dilemma. We give a scheduling method for regularizer coefficient through ablation experiments. Furthermore, we adapt our regularizer into several differentiable NAS algorithms on different datasets and different search spaces, achieving significant improvements, i.e., higher accuracy and better robustness. We report the best and average errors of our searched architectures in different runs with different seeds in Tables 2 and 3. The best architecture achieves 2.49% test error on CIFAR-10 and 24.0% top-1 error when transferred to ImageNet, which are highly competitive results.

Our contributions can be summarized as follows:

- We argue that the failure of architecture parameters learning is the fundamental cause of the Matthew effect and discretization discrepancy for DARTS. And we introduce the architecture entropy to measure the discrepancy of the architecture parameters of different candidate operations.
- We propose an architecture entropy regularizer, which avoids the unfair learning of network parameters caused by the Matthew effect in the early search phase and the performance drop caused by discretization discrepancy after the search phase. It can be easily adapted in different differentiable NAS algorithms with extremely little cost of time and memory.
- Extensive experiments and ablation studies prove that our architecture entropy regularizer can significantly improve different differentiable NAS algorithms on different datasets and different search spaces. We report the best architecture we discovered and its competitive results on CIFAR-10 and ImageNet. The code is available at https://github.com/kunjing96/DARTS-AER.

2. Related work

Since DARTS (Liu et al., 2019) is proposed, the differentiable methods have been the mainstream of neural architecture search. Many researchers proposed the same defects of DARTS as ours. They also suggested many amazing improvements for these problems. PDARTS (Chen et al., 2019) gradually increases the network depth during the search phase (totally 3 stages) to alleviate a significant optimization gap that is caused by the different properties of the search and evaluation phases. This gap is also known as discretization discrepancy, which makes the architecture parameters overfit the super-network. For each stage of PDARTS, only network parameters are tuned in the first 10 epochs while both network and architecture parameters are jointly optimized in the rest 15 epochs, which alleviates the rich-get-richer problem. FairNAS (Chu, Zhang, & Xu, 2021) and Fair DARTS (Chu, Zhou et al., 2020) also found the Matthew effect in DARTS, which was called the unfair training or the unfair advantage in an exclusive competition. They argue that the unfair advantage makes DARTS suffer from well-known performance collapse. Besides, other works (Hong et al., 2020; Li et al., 2019; Liang et al., 2019; Yu, Sciuto, Jaggi, Musat, & Salzmann, 2020; Zheng et al., 2020) came up with similar proposals as well. Although observing the same findings as them, we argue that the solutions to these two problems are inconsistent, and propose an easier way to escape from the dilemma.

Implicit and explicit regularization methods have been widely used in NAS. For example, in order to search for efficient neural architecture, extensive researches (Cai, Zhu, & Han, 2019; Green, Vineyard, Helinski, & Koç, 2020; Wu et al., 2019; Xie, Zheng, Liu, & Lin, 2019; Zhang, Yang, Jiang, Zhu, & Liu, 2020) regard forwarding time or occupied hardware resources of the child network as a regularizer in the search objective, which adjusts the sizes of the searched architectures through the regularizer coefficient. SmoothDARTS (Chen & Hsieh, 2020) proposes a perturbation-based regularization to smooth the loss landscape and improves the generalizability of differentiable NAS algorithms. RDARTS (Zela et al., 2020) shows that properly regularizing the inner objective (e.g., regularization via data augmentation and L_2 regularization) can help to control the eigenvalues of the Hessian matrix of the validation loss and therefore improves generalization. Besides, there are many implicit regularizations (Chu, Zhang, & Li, 2020; Chu, Zhou et al., 2020; Liang et al., 2019; Zela et al., 2020), including early stopping, adding noise, and restricting the number of skip connections.

In particular, we also find that the two related works (Ferianc, Fan, & Rodrigues, 2020; Gao et al., 2020) also use a regularizer similar to our proposal. MTL-NAS (Gao et al., 2020) proposes a single-shot gradient-based search algorithm, which closes the performance gap between the searched architectures and the final evaluation architecture by a minimum entropy regularization on the architecture weights during the search phase. This makes the architecture weights converge to near-discrete values. As a result, the searched model can be directly used for evaluation without (re-)training from scratch. VINNAS (Ferianc et al., 2020) presents a differentiable variational inference-based NAS method for searching sparse convolutional neural networks by gradually removing unnecessary operations and connections. To achieve a high level of certainty in the selection of operations, they minimize the joint entropy across the potential operations. Although we also use the entropy of architecture parameters as the regular term in the search process, we aim to solve the above dilemma of the Matthew effect and discretization discrepancy. Specifically, we define the average entropy of all architecture parameters as architecture entropy, argue that architecture entropy regularizer has two effects, and regularize the learning of architecture parameters in different ways during different search stages.

3. Escaping from dilemma

3.1. Differentiable architecture search

We follow the DARTS framework (Liu et al., 2019), which aims to search for two types of cells (including a normal cell and a reduction cell) that can be stacked to form the optimal target network. Each cell is defined as directed acyclic graph (DAG) of N nodes, where each node $x^{(i)}$ is a latent representation and each edge (i, j) is associated with a set of candidate operations $o^{(i,j)}$. We denote the candidate operation space as $\mathcal{O}$, which is exactly the same as the original DARTS setting.

DARTS relaxes the discrete search space to be continuous by a weighted sum of all candidate operations, which can be formulated as

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

The architecture weight $p^{(i,j)}$ for an edge (i, j) is parameterized by a vector (called architecture parameter) $\alpha^{(i,j)}$ of dimension $|\mathcal{O}|$. Intuitively, the architecture parameter $\alpha^{(i,j)}$ represents the relative contribution of the operation $o^{(i,j)}$ for transforming the feature map $x^{(i)}$.

After the relaxation, the task of neural architecture search then reduces to learning a set of continuous variables $\alpha = \{\alpha^{(i,j)}\}$. Specifically, DARTS formulates the jointly learning of the architecture parameters α and the network parameters ω within all the mixed operations as a bi-level optimization problem:

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

DARTS approximates $\omega^*(\alpha)$ by adapting ω using only a single gradient descent step and updates ω and α alternatively using the training and validation set respectively.

At the end of search, we can derive the final searched architecture by (1) replacing each mixed operation $\bar{o}^{(i,j)}$ with the most likely operation $o^{(i,j)} = \text{argmax}_{o\in\mathcal{O}, o\neq zero} p_o^{(i,j)}$ and (2) retaining two edges from different predecessor nodes with the largest $\max_{o\in\mathcal{O}, o\neq zero} p_o^{(i,j)}$ for each intermediate node.

3.2. Dilemma of differentiable architecture search

Observing the search process of DARTS and its discovered architecture, we discover that two problems make DARTS perform poorly, i.e., the Matthew effect and discretization discrepancy.

Matthew effect. We check the search process of DARTS and the discovered architectures and find that at the beginning of the search, the non-parameterized operations often perform better because the parameterized operations have not learned any knowledge. Besides, some candidate operations may learn to express the desired functions more quickly. This makes them learn larger architecture weights and gain larger gradients of network parameters, which accelerates the parameter update of the correlative operations. It is a vicious circle that the premature dominant expression can cause unfair training and further strengthen the dominant expression, thus ignoring the expression of these disadvantaged operations at the beginning. This phenomenon is summarized as the rich-get-richer, also known as the Matthew effect.

Discretization discrepancy. During the search phase of DARTS, continuous architecture parameters α are used for the relaxation. However, they have to be discretized to derive the searched architecture eventually. This requires that a better mixture of architectures implies a better final architecture, but it is not the case. We observe that there is always a large performance discrepancy in the final discretization phase of DARTS. In other words, the validation error reduction of the mixture of architectures is not always related to the validation error reduction of the final architecture. At the end of the search, the architecture weight seriously deviating from the one-hot vector we expect causes this phenomenon, named discretization discrepancy.

To alleviate the Matthew effect, we expect that the architecture weight $p_o^{(i,j)}$ for each candidate operation of edge (i, j) can be close enough so that their network parameters can be updated using fair gradients. But for relieving the discretization discrepancy, we require each architecture weight vector to be a one-hot vector, making the architecture weight of a certain candidate operation prominent, i.e., the dominant expression. These two aspects make DARTS into a dilemma.

3.3. Architecture entropy regularizer

In information theory, entropy is the measurement of uncertainty. With the decrease of entropy, random events become more definite. Naturally, we use entropy $H(\alpha^{(i,j)}) = -\sum_{o\in\mathcal{O}} p_o^{(i,j)} \cdot \log_2 p_o^{(i,j)}$ to measure the discrepancy of the architecture parameters $\alpha_o^{(i,j)}$ for different candidate operations of each edge (i, j). We define the mean entropy $\bar{H}(\alpha) = \frac{1}{N}\sum_{(i,j)} H(\alpha^{(i,j)})$ of all edges as the architecture entropy, which reflects the learning of the whole architecture parameters.

Furthermore, we introduce the architecture entropy regularizer (AER) to control the learning of architecture parameters. Therefore, the bi-level optimization of NAS is rewritten as

$$\begin{aligned} &\min_{\alpha} \quad \mathcal{L}_{val}(\omega^*(\alpha), \alpha) + \lambda \cdot \bar{H}(\alpha) \\ &s.t. \quad \omega^*(\alpha) = \text{argmin}_{\omega}\ \mathcal{L}_{train}(\omega, \alpha). \end{aligned} \tag{3}$$

3.3.1. Two effects

Obviously, the optimization defined by DARTS in Formula (2) is a special case of our definition in Formula (3). When $\lambda = 0$, the optimization is equivalent to the definition of DARTS. As shown in Figs. 2 and 3, our architecture entropy regularizer can achieve two different effects by the different setups of its coefficient.

When the coefficient $\lambda < 0$, the architecture parameters are learned in the direction of increasing architecture entropy, as shown in Fig. 6(a). In particular, the architecture entropy is large initially because the architecture parameters of the candidate operations are randomly initialized with Gaussian distribution. Therefore, it is hard to increase, but remains unchanged or slowly declines as far as possible. Compared with Row 1 ($\lambda = 0$) and Row 2 ($\lambda < 0$) in Fig. 2, the weights of different candidate operations of the latter are closer. All the candidate operations are treated fairly, which alleviates the Matthew effect. Compared with Column 1 ($\lambda = 0$) and Column 2 ($\lambda < 0$) in Fig. 3, after alleviating the Matthew effect, we can gain a better evaluation ranking with higher Kendall τ and lower average error.

When the coefficient $\lambda > 0$, the architecture parameters are learned in the direction of decreasing architecture entropy, as shown in Fig. 6(a). This accelerates the dominant expression of the optimal candidate operation and improves the search efficiency. On the other hand, at the end of searching as shown in Row 3 of Fig. 2, the architecture weight $p^{(i,j)}$ of each edge (i, j) will

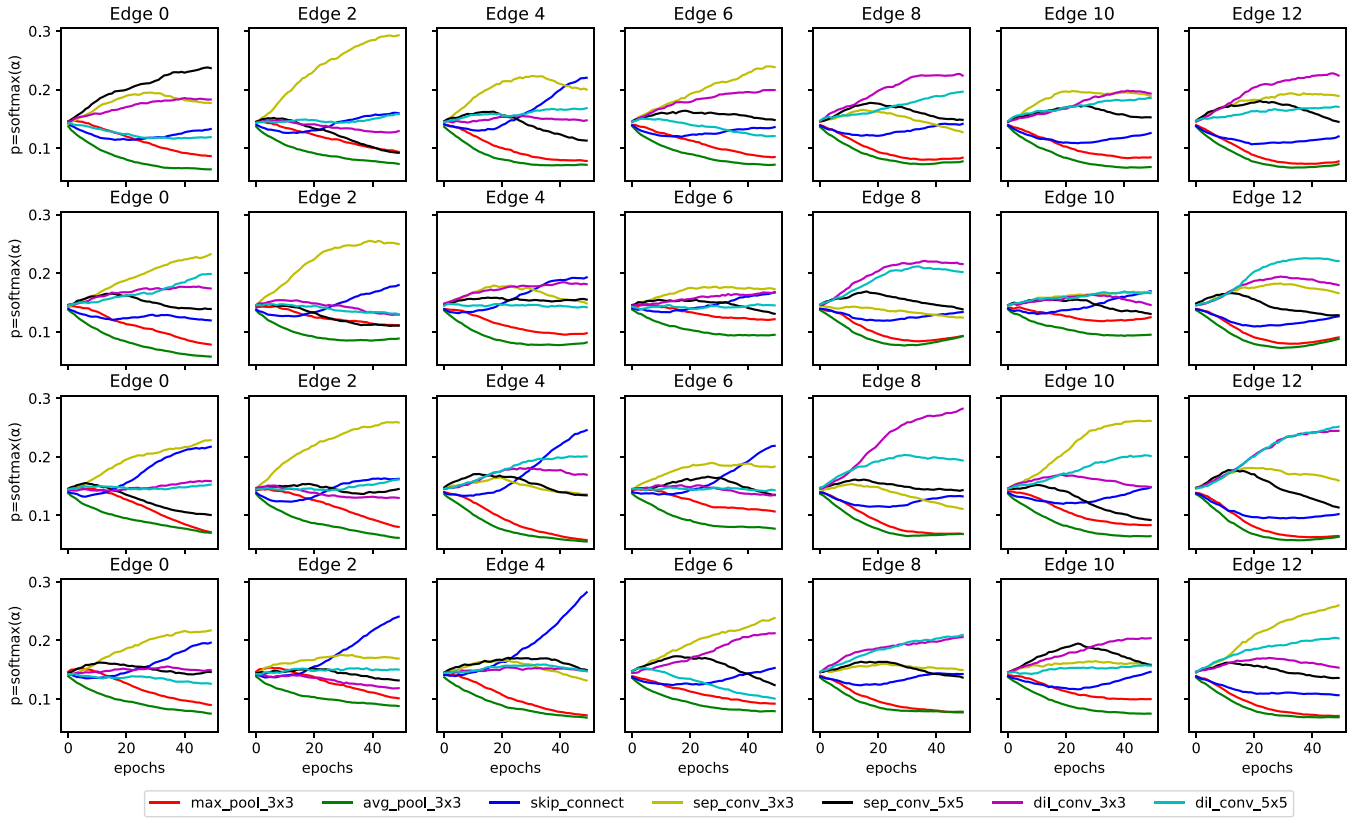


Fig. 2. Architecture weight evolutions during the search phase with different regularizer coefficients $\lambda(t)$. Due to the limited space, we just show the evolutions on edge of even index for one run. Row 1: $\lambda(t) = 0$ (DARTS). Row 2: $\lambda(t) = -0.2$. Row 3: $\lambda(t) = 0.2$. Row 4: $\lambda(t)$ changes from -0.2 to 0.2 with epochs following Fig. 1. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

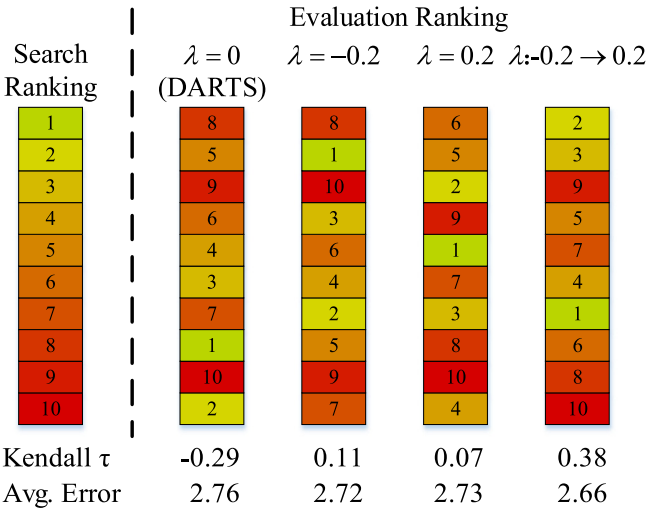


Fig. 3. The search-evaluation architecture rankings for different ways of regularizer coefficient $\lambda(t)$ scheduling. We show Kendall τ correlations and average test error on CIFAR-10. Architectures are obtained from 10 independent runs of search and evaluation with different seeds.

be closer to the one-hot vector, which indicates less discretization discrepancy. Compared with Column 1 ($\lambda = 0$) and Column 3 ($\lambda > 0$) in Fig. 3, we can also gain a better evaluation. However, it aggravates the Matthew effect in the early stage of the search. As shown in Row 3 ($\lambda > 0$) of Fig. 2, the dominant expression of candidate operations becomes more significant than Row 1 ($\lambda = 0$) in the early stage of the search.

3.3.2. Regularizer coefficient scheduling

We expect a later dominant expression for the Matthew effect and an earlier dominant expression for discretization discrepancy, which are completely contradictory. For the trade-off in the dilemma, we propose that the architecture parameters are learned to prevent the dominant expression in the early search stage and promote the dominant expression in the later search stage by scheduling the regularizer coefficient $\lambda(t)$ as shown in Fig. 1. The regularizer coefficient $\lambda(t)$ increases from λ_{neg} to λ_{pos} following a cosine schedule, where λ_{neg} and λ_{pos} are negative and positive respectively. The optimization formula is rewritten as

$$\begin{aligned} \min_{\alpha} \quad & \mathcal{L}_{val}(\omega^*(\alpha), \alpha) + \lambda(t) \cdot \bar{H}(\alpha) \\ s.t. \quad & \omega^*(\alpha) = \operatorname{argmin}_{\omega} \mathcal{L}_{train}(\omega, \alpha). \end{aligned} \tag{4}$$

As shown in Row 4 of Fig. 2, when λ changes from λ_{neg} to λ_{pos} with the training epochs, each architecture weight tends to be the one-hot vector from the initial balanced state. Fig. 3 shows that the regularizer with the coefficient scheduling can significantly improve the evaluation ranking. The scheduling is proved to be effective by extensive experiments in Section 4.

3.4. Search algorithms

The optimization algorithm is described in Algorithm 1. Following DARTS, we update architecture parameters α and network parameters ω alternately.

4. Experiments

We conduct experiments on one NAS benchmark NAS-Bench-201 (Dong & Yang, 2020) and two popular image classification

Table 1
Comparison of different algorithms on NAS-Bench-201. The first block shows results of parameter sharing based NAS methods provided by the benchmark.

Algorithm	Cost	CIFAR-10 Acc (%)		CIFAR-100 Acc (%)		ImageNet16-120 Acc (%)	
	(s)	Validation	Test	Validation	Test	Validation	Test
RSPS[a]	8007	80.42 ± 3.58	84.07 ± 3.61	52.12 ± 5.55	52.31 ± 5.77	27.22 ± 3.24	26.28 ± 3.09
DARTS-V1[a]	11625	39.77 ± 0.00	54.30 ± 0.00	15.03 ± 0.00	15.61 ± 0.00	16.43 ± 0.00	16.32 ± 0.00
DARTS-V2[a]	35781	39.77 ± 0.00	54.30 ± 0.00	15.03 ± 0.00	15.61 ± 0.00	16.43 ± 0.00	16.32 ± 0.00
GDAS[a]	31609	89.89 ± 0.08	93.61 ± 0.09	71.34 ± 0.04	70.70 ± 0.30	41.59 ± 1.33	41.71 ± 0.98
SETN[a]	34139	84.04 ± 0.28	87.64 ± 0.00	58.86 ± 0.06	59.05 ± 0.24	33.06 ± 0.02	32.52 ± 0.21
ENAS[a]	14058	37.51 ± 3.19	53.89 ± 0.58	13.37 ± 2.35	13.96 ± 2.33	15.06 ± 1.95	14.84 ± 2.10
DARTS-V1[a]	11668	39.77 ± 0.00	54.30 ± 0.00	15.03 ± 0.00	15.61 ± 0.00	16.43 ± 0.00	16.32 ± 0.00
DARTS-V1-AER[a]	11699	**73.88 ± 7.91**	**76.68 ± 8.15**	**46.42 ± 11.11**	**46.53 ± 10.69**	**24.78 ± 8.36**	**24.48 ± 8.57**
DARTS-V1[b]	25031	77.69 ± 6.65	79.75 ± 6.24	49.24 ± 7.54	49.42 ± 7.29	24.40 ± 3.91	23.54 ± 3.62
DARTS-V1-AER[b]	24174	**82.39 ± 0.00**	**84.16 ± 0.00**	**54.57 ± 0.00**	**54.64 ± 0.00**	**27.17 ± 0.00**	**26.10 ± 0.00**
DARTS-V1[c]	62041	83.65 ± 2.84	85.90 ± 3.24	56.51 ± 5.35	57.21 ± 5.12	29.03 ± 5.70	27.73 ± 4.66
DARTS-V1-AER[c]	63051	**85.44 ± 0.95**	**87.98 ± 0.91**	**60.13 ± 2.05**	**60.22 ± 1.49**	**32.83 ± 1.07**	**31.14 ± 1.20**
DARTS-V2[a]	33298	39.77 ± 0.00	54.30 ± 0.00	15.03 ± 0.00	15.61 ± 0.00	16.43 ± 0.00	16.32 ± 0.00
DARTS-V2-AER[a]	36992	**68.29 ± 0.00**	**70.92 ± 0.00**	**38.57 ± 0.00**	**38.97 ± 0.00**	**18.87 ± 0.00**	**18.42 ± 0.00**
DARTS-V2[b]	65746	68.29 ± 0.00	70.92 ± 0.00	38.57 ± 0.00	38.97 ± 0.00	18.87 ± 0.00	18.41 ± 0.00
DARTS-V2-AER[b]	67643	**83.45 ± 1.15**	**85.47 ± 1.94**	**56.73 ± 3.60**	**57.07 ± 3.15**	**29.52 ± 3.20**	**28.38 ± 3.12**
DARTS-V2[c]	177515	83.86 ± 1.14	86.58 ± 1.75	58.43 ± 2.89	58.71 ± 2.84	30.67 ± 2.76	30.03 ± 3.14
DARTS-V2-AER[c]	177821	**85.37 ± 1.20**	**87.51 ± 1.35**	**59.90 ± 2.80**	**60.14 ± 2.61**	**31.20 ± 2.46**	**30.21 ± 2.83**
GDAS[a]	31502	90.01 ± 0.10	93.57 ± 0.15	71.12 ± 0.35	70.73 ± 0.25	41.16 ± 0.71	42.22 ± 0.29
GDAS-AER[a]	34072	89.71 ± 0.04	93.46 ± 0.21	70.42 ± 0.50	**71.09 ± 0.26**	40.39 ± 0.17	41.54 ± 0.70
GDAS[b]	62274	88.57 ± 0.12	92.23 ± 0.07	67.34 ± 0.37	67.33 ± 0.10	39.22 ± 0.03	39.28 ± 0.41
GDAS-AER[b]	64633	**89.38 ± 0.03**	**92.82 ± 0.51**	**69.21 ± 0.30**	**69.78 ± 0.41**	**41.89 ± 1.16**	**42.12 ± 0.53**
GDAS[c]	161535	89.91 ± 0.03	93.50 ± 0.23	70.81 ± 0.79	70.67 ± 0.34	40.47 ± 0.27	40.64 ± 0.54
GDAS-AER[c]	172525	**90.17 ± 0.16**	93.37 ± 0.07	70.15 ± 0.86	70.35 ± 0.41	**42.12 ± 1.63**	**42.26 ± 1.20**

[a]The last three blocks show the our results of DARTS (with first-order and second-order approximation, marked by V1 and V2 respectively) and GDAS on three different datasets (CIFAR-10).
[b]The last three blocks show the our results of DARTS (with first-order and second-order approximation, marked by V1 and V2 respectively) and GDAS on three different datasets (CIFAR-100).
[c]The last three blocks show the our results of DARTS (with first-order and second-order approximation, marked by V1 and V2 respectively) and GDAS on three different datasets (ImageNet16-120).
Our architecture entropy regularizer is used for their search (marked by AER, i.e., DARTS-V1-AER, DARTS-V2-AER, and GDAS-AER). We report the mean and standard deviation of accuracies of 3 runs. The search cost is converted to the search cost on a single GeForce GTX 1080 Ti GPU so that it can be compared with the results provided by NAS-Bench-201.

Algorithm 1 Differentiable Architecture Search with Architecture Entropy Regularizer

Create a mixed operation $\bar{o}^{(i,j)}$ parameterized by $\alpha^{(i,j)}$ for each edge (i, j)
while not converged **do**
Update architecture α by descending

$$\nabla_\alpha \mathcal{L}_{val}(\omega - \xi \nabla_\omega \mathcal{L}_{train}(\omega, \alpha), \alpha) + \lambda(t) \cdot \bar{H}(\alpha)$$

($\xi = 0$ if using first-order approximation)
Update weights ω by descending

$$\nabla_\omega \mathcal{L}_{train}(\omega, \alpha)$$

end while
Derive the final architecture based on the learned α

datasets, i.e., CIFAR-10 (Krizhevsky & Hinton, 2009) and ImageNet (Deng et al., 2009).

4.1. Experiments on NAS-Bench-201

NAS-Bench-201. NAS-Bench-201 (Dong & Yang, 2020) is a purpose-built benchmark for prototyping NAS algorithms. It contains 15,625 CNN models from a fixed cell-based search space and corresponding detailed training log (including performance metrics and diagnostic information) on three different datasets, i.e., CIFAR-10 (Krizhevsky & Hinton, 2009), CIFAR-100 (Krizhevsky & Hinton, 2009), and ImageNet16-120 (Chrabaszcz, Loshchilov, & Hutter, 2017).

Implementation details. To verify our method on the different search spaces, we conduct experiments based on two typical NAS algorithms, i.e., DARTS (Liu et al., 2019) and GDAS (Dong & Yang, 2019), on the NAS-Bench-201 search space. In the search phase, we directly use the code provided by NAS-Bench-201 (Dong & Yang, 2020). Then we obtain the training information of the architectures from the look-up table in NAS-Bench-201. All experimental configurations exactly follow the corresponding works (Dong & Yang, 2019, 2020; Liu et al., 2019). We run each algorithm three times. All experiments are run on a single Tesla V100.

Results analysis. As shown in Table 1, the performances of our DARTS-AER discovered architectures on the three datasets are significantly better than those of the original DARTS. In the third block, the most remarkable row is DARTS-V1-AER when searching on CIFAR-10, where the test accuracies of the three datasets are improved by 22.38%, 30.92%, 8.61% respectively. Meanwhile, although our GDAS-AER does not perform well on CIFAR-10, its discovered architectures have the higher accuracies when searching on CIFAR-100 and ImageNet with lower standard deviation. Our GDAS-AER algorithm achieves the best 42.26% average test accuracy on ImageNet. This demonstrates our proposed architecture entropy regularizer helps DARTS (first-order and second-order) and GDAS algorithms to find those better architectures and improves them to be more robust. Experimental results confirm our architecture entropy regularizer can improve different differentiable NAS algorithms to perform better on different search spaces and different datasets.

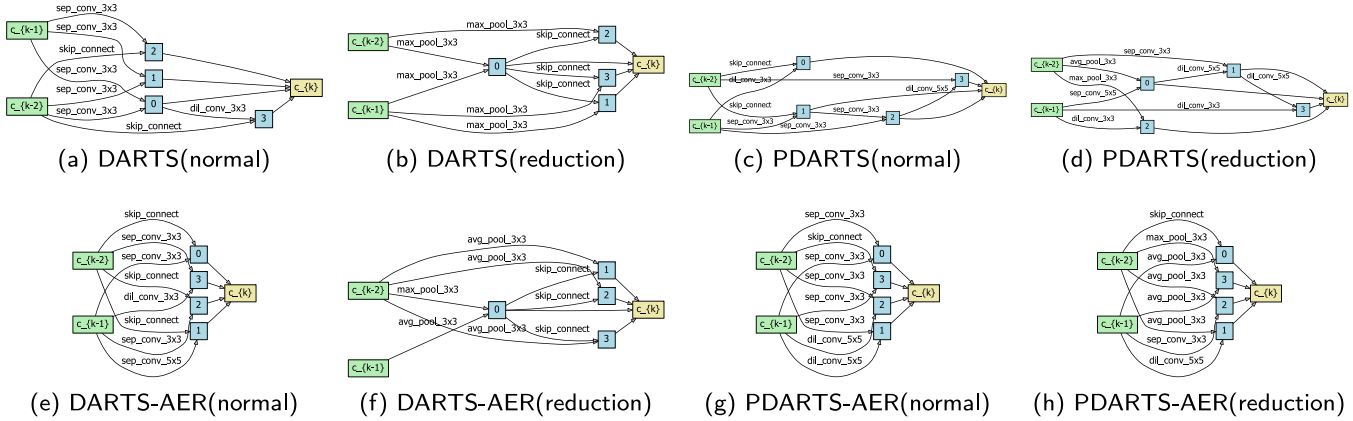


Fig. 4. The discovered architectures on CIFAR-10. (normal) denotes the normal cell of these architectures; (reduction) denotes the reduction cell of these architectures.

4.2. Experiments on CIFAR-10

CIFAR-10. CIFAR-10 (Krizhevsky & Hinton, 2009) has 50K/10K training/testing images with a fixed spatial resolution of 32×32. These images are equally distributed over 10 classes. In the search phase, the training set is equally split into two subsets, one for tuning network parameters and the other for tuning architecture parameters. In the evaluation phase, the standard training/testing split is used.

4.2.1. Architecture search

Implementation details. To verify the validity of our method, we conduct experiments based on two typical NAS algorithms, i.e., DARTS (Liu et al., 2019) and PDARTS (Chen et al., 2019). In the search phase, we directly use their approaches except for adding our architecture entropy regularizer. Since we use DARTS and PDARTS as the backbone algorithm, the search space and hyperparameter settings are exactly the same as theirs for a fair comparison. Both of them aim to search for two types of cells, a normal cell and a reduction cell. The cells are represented by a DAG of 7 nodes with each edge having 8 candidate operations, consisting of 3×3 and 5×5 separable convolutions, 3×3 and 5×5 dilated separable convolutions, 3×3 max pooling, 3×3 average pooling, identity, and zero. In the search phase, we set $|\lambda_{neg}| = |\lambda_{pos}| = 0.2$, i.e., $\lambda_{neg} = -0.2$ and $\lambda_{pos} = 0.2$. The ablation study in Section 4.4.2 shows that this is a good scheduling way. We run each algorithm three times. All experiments are run on a single Tesla V100.

Results analysis. Our discovered architectures on CIFAR-10 are shown in Fig. 4. Compared to the cells discovered by DARTS and PDARTS, there are some distinct differences in our architectures. One difference is that our architectures are shallower than DARTS and PDARTS. Another difference is that our algorithm prefers to select 3×3 average pooling instead of 3×3 max pooling in the reduction cell. Besides, as shown in Table 2, our algorithm discovers architectures with fewer parameters than DARTS and PDARTS. We argue that we can discover the architecture with shallower depth and fewer parameters because the search is carried out on the small dataset CIFAR-10 without considering transferability and this architecture can meet the needs of small datasets. We think that increasing the depth of cells is not always useful. Although, the depth of network is positively correlated with the model expression ability to a certain extent, it cannot reflect the generalization performance given a dataset and a set of hyper-parameters. Our shallower architectures also achieve lower errors over different datasets, which is due to easier training and less generalization error. These differences may bring some inspiration to the design of neural architecture.

4.2.2. Architecture evaluation

Implementation details. In the evaluation phase, we stack an evaluation network with 20 repeated cells and 36 initial channels. The evaluation network is trained on CIFAR-10 from scratch for 600 epochs. Additional enhancements are applied during the evaluation phase, including cutout regularization of length 16, drop-path of probability 0.3, and auxiliary towers with weight 0.4. We adopt a standard SGD optimizer with a weight decay of 0.0003 and a momentum of 0.9. The initial learning rate is set to 0.025, decreasing to 0 following a cosine scheduling. All configurations remain the same as the settings of DARTS (Liu et al., 2019) and PDARTS (Chen et al., 2019). All experiments are run on a single Tesla V100.

Results analysis. As shown in Table 2, the average test error of DARTS-AER is 0.1% lower than that of DARTS implemented by us and the standard deviation of DARTS-AER is 0.07 lower. Similarly, compared with our implemented PDARTS, the average test error of PDARTS-AER is 0.07% lower and the standard deviation of PDARTS-AER is 0.02 lower. The evaluation results show that our discovered architectures outperform the architectures searched by those algorithms without the architecture entropy regularizer. This demonstrates that our architecture entropy regularizer promotes different differentiable NAS algorithms to obtain better architectures. We achieve a highly competitive result (average test error of 2.49%) on CIFAR-10. The lower standard deviation also shows that our architecture entropy regularizer can improve the robustness of those differentiable NAS algorithms. Notably, the regularizer introduces no extra significant cost of time and memory.

4.2.3. Architecture transferability evaluation

Implementation details. We evaluate the transferability of searched architectures on ImageNet as well. The best architecture we discovered on CIFAR-10 is trained. We build an evaluation network with 14 cells and 48 initial channels. The network is trained for 250 epochs with batch size 128. An SGD optimizer with a weight decay of 3×10^{-5}, a momentum of 0.9, and an initial learning rate of 0.1 (annealed down to zero following a cosine schedule without restart) is used to optimize the network parameters. Other training tricks are composed of label smoothing, auxiliary tower, and learning rate warmup during training. Other hyperparameters exactly follow PDARTS (Chen et al., 2019). All experiments are run on a single Tesla V100.

Results analysis. The results of architecture transferability evaluation and comparison with other NAS algorithms are illustrated in Table 3. Our discovered architecture achieves a 24.3% top-1 error with 5.2M parameters and 587M FLOPs when transferred

Table 2
Results of different algorithms on CIFAR-10. Our experimental results are the average test error obtained by running our algorithms (including the search and evaluation phase) three times.

Algorithm	Test error (%)		Params	Search cost	Search method
	Best	Average	(M)	(GPU days)	
DenseNet-BC (Huang, Liu, van der Maaten, & Weinberger, 2017)[a]	3.46	–	25.6	–	Manual
NASNet-A (Zoph et al., 2018)	2.65	–	3.3	1800	RL
AmoebaNet-B (Real, Aggarwal, Huang, & Le, 2019)	2.55	–	2.8	3150	Evolution
ENAS (Pham et al., 2018)	2.89	–	4.6	0.5	RL
NAONet-WS (Luo, Tian, Qin, Chen, & Liu, 2018)	2.93	–	2.5	0.2	NAO
SNAS (Xie et al., 2019)	2.85	–	2.8	1.5	Gradient
GDAS (Dong & Yang, 2019)	2.93	–	3.4	0.21	Gradient
BayesNAS (Zhou, Yang, Wang, & Pan, 2019)	2.81	–	3.4	0.2	Gradient
PCDARTS (Xu et al., 2020)	2.57	–	3.6	0.1	Gradient
DropNAS (Hong et al., 2020)	2.26	2.58 ± 0.14	4.1	0.6	Gradient
NASP (Yao, Xu, Tu, & Zhu, 2020)	2.83	–	3.3	0.1	Gradient
SDARTS-ADV (Chen & Hsieh, 2020)	2.61	–	3.3	1.3	Gradient
PDARTS-ADV (Chen & Hsieh, 2020)	2.48	–	3.4	1.1	Gradient
SGAS (Li et al., 2020)	2.39	2.66 ± 0.24	3.7	0.25	gradient
FairDARTS (Chu, Zhou et al., 2020)	–	2.54 ± 0.05	3.32 ± 0.46	0.42	Gradient
DARTS (Liu et al., 2019)	2.76	–	3.3	4	Gradient
DARTS[b]	2.63	2.76 ± 0.12	3.59 ± 0.35	4	Gradient
DARTS-AER[b]	**2.60**	**2.66 ± 0.05**	**3.39 ± 0.14**	4	Gradient
PDARTS (Chen et al., 2019)	2.50	–	3.4	0.3	Gradient
PDARTS[b]	2.54	2.60 ± 0.04	3.74 ± 0.19	0.3	Gradient
PDARTS-AER[b]	**2.49**	**2.53 ± 0.02**	**3.64 ± 0.18**	0.3	Gradient

[a]denotes training without cutout augmentation.
[b]denotes that it is implemented by us.
Different from that other NAS algorithms (like DARTS, PDARTS, and PCDARTS) pick the best one, we report the mean and standard deviation over 3 single runs. The search cost is converted so that it can be compared with the corresponding algorithm.

to ImageNet, which outperforms DARTS (Liu et al., 2019) and PDARTS (Chen et al., 2019). Experimental results prove that our architecture has better transferability.

4.3. Experiments on ImageNet

ImageNet. In Section 4.2.3, we use ILSVRC2012 (ImageNet) (Russakovsky et al., 2015) to evaluate the transferability of the architectures discovered on CIFAR-10. In this section, we search for the optimal architecture on ILSVRC2012 directly. ILSVRC2012 is a subset of ImageNet which contains 1000 object categories and 1.28M training and 50K validation images. We apply the original dataset split setting where the input image size is 224×224 for architecture evaluation. For architecture search, we use ImageNet16-120 dataset (Chrabaszcz et al., 2017), which is built from the down-sampled variant of ImageNet (ImageNet16 $\times$ 16). ImageNet16-120 contains 151.7K training images, 3K validation images, and 3K test images with 120 classes.

4.3.1. Architecture search

Implementation details. Strictly following the configuration of PDARTS (Chen et al., 2019), we use a similar configuration to the one used on CIFAR10, except for some minor changes. For the search stage 1, 2, and 3, we respectively set the number of cells to 5, 8, 11, the dropout rate to 0.0, 0.3, 0.6, and the number of initial channels to 16, 28, 40. All experiments are run on a single Tesla V100.

Results analysis. Our discovered architectures on ImageNet are shown in Fig. 5. Compared with the discovered architecture on CIFAR-10, the discovered architecture has more deep connections. This is because the complexity of ImageNet makes those shallow architectures more difficult to fit, forcing the algorithm to explore deeper architectures. Meanwhile, the larger dataset makes the deep network easier to train and can reduce generalization error, which also makes the algorithm explore deeper architectures. Moreover, the proportion of 3×3 average pooling in the reduction cell remains higher than other architectures.

4.3.2. Architecture evaluation

Implementation details. We evaluate the discovered architecture on ImageNet with the same configure in Section 4.2.3. All experiments are run on a single Tesla V100.

Results analysis. As shown in Table 3, our discovered architecture on ImageNet achieves a 24.0% top-1 error with 5.1M parameters and 578M FLOPs, which outperforms all the other NAS algorithms mentioned. Experimental results demonstrate our algorithm can perform well on different datasets.

4.4. Ablation studies

4.4.1. Ablation study for architecture entropy regularizer

We conduct an ablation study on CIFAR-10. We assemble the architecture entropy regularizer for DARTS, and all the settings are the same as in Sections 4.2.1 and 4.2.2. We conduct experiments in four cases: $\lambda(t) = 0$ that is equivalent to DARTS, $\lambda(t) = -0.2$, $\lambda(t) = 0.2$, and $\lambda(t)$ from -0.2 to 0.2. We run the search and evaluation ten times for each case.

As we can see in Fig. 3, the rank correlation is stronger when $\lambda(t) = 0.2$ and $\lambda(t)$ varies from -0.2 to 0.2 than when $\lambda(t) = 0$ and $\lambda(t) = -0.2$. Simultaneously, the average error is lower when $\lambda(t)$ varies from -0.2 to 0.2. This shows that our architecture entropy regularizer with a positive coefficient can alleviate the discretization discrepancy problem and the method with regularizer coefficient scheduling outperforms others. Fig. 2 illuminates that for the last case, the curve in the early stage tends to the trend of the second case and in the later stage tends to the trend

Table 3
Results of different algorithms on ImageNet.

Algorithm	Test error (%)		Params	+×	Search cost	Search method
	Top-1	Top-5	(M)	(M)	(GPU days)	
MobileNet-V2 (Sandler, Howard, Zhu, Zhmoginov, & Chen, 2018)	25.3	–	6.9	585	–	Manual
ShuffleNet-V2 (Ma, Zhang, Zheng, & Sun, 2018)	25.1	–	7.4	591	–	Manual
EfficientNet-B0 (Tan & Le, 2019)[c]	23.7	6.8	5.3	390	–	Manual
NASNet-A (Zoph et al., 2018)	26.0	8.4	5.3	564	1800	RL
AmoebaNet-C (Real et al., 2019)[b]	24.3	7.6	6.4	570	3150	Evolution
PNAS (Liu, Zoph, Neumann, Shlens, Hua, Li et al., 2018)	25.8	8.1	5.1	588	∼225	SMBO
MnasNet-92 (Tan et al., 2019)[bc]	25.2	8.0	4.4	388	–	RL
DARTS (Liu et al., 2019)	26.7	8.7	4.7	574	4	Gradient
FBNet-C (Wu et al., 2019)[ab]	25.1	7.7	5.5	375	9	Gradient
ProxylessNAS (Cai et al., 2019)[ab]	24.9	7.5	7.1	465	8.3	Gradient
SNAS (Xie et al., 2019)	27.3	9.2	4.3	522	1.5	Gradient
GDAS (Dong & Yang, 2019)	26.0	8.5	5.3	581	0.21	Gradient
NASP (Yao et al., 2020)	27.2	9.1	4.6	–	0.1	Gradient
BayesNAS (Zhou et al., 2019)	26.5	8.9	3.9	–	0.2	Gradient
PCDARTS (Xu et al., 2020)	25.1	7.8	5.3	586	0.1	gradient
Single-Path NAS (Stamoulis et al., 2019)[ab]	25.0	7.8	–	–	0.16	Gradient
PDARTS-ADV (Chen & Hsieh, 2020)	24.2	7.2	–	–	1.1	Gradient
FairDARTS-B (Chu, Zhou et al., 2020)[b]	24.9	7.5	4.8	541	0.4	Gradient
FairDARTS-D (Chu, Zhou et al., 2020)[ab]	24.4	7.4	4.3	440	3	Gradient
PDARTS (Chen et al., 2019)	24.4	7.4	4.9	557	0.3	gradient
PDARTS-AER[d]	**24.3**	**7.2**	5.2	587	0.3	Gradient
PDARTS[a] (Chen et al., 2019)	24.1	7.3	5.4	597	2.0	gradient
PDARTS-AER[ad]	**24.0**	**7.2**	**5.1**	**578**	2.0	Gradient

[a]denotes directly searching on ImageNet.
[b]denotes using a different search space from ours.
[c]denotes training with more tricks, like AutoAugment, etc.
[d]denotes that it is implemented by us.
The search cost is converted so that it can be compared with the corresponding algorithm.

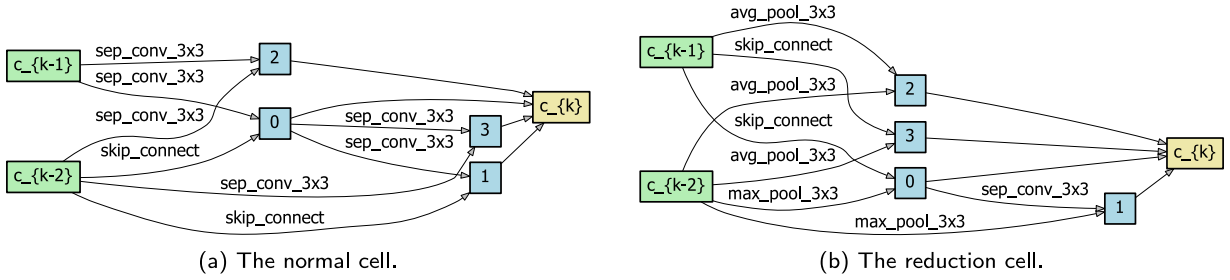


Fig. 5. The discovered architecture by PDARTS-AER on ImageNet.

of the third case, which proves that when $\lambda(t)$ varies from -0.2 to 0.2, the algorithm integrates the advantages of the second and third cases, i.e., $\lambda(t) = -0.2$ and $\lambda(t) = 0.2$. In brief, our architecture entropy regularizer achieves two different effects by the different setups of its coefficient, and the regularizer coefficient scheduling promotes the combination of the two effects to perform better.

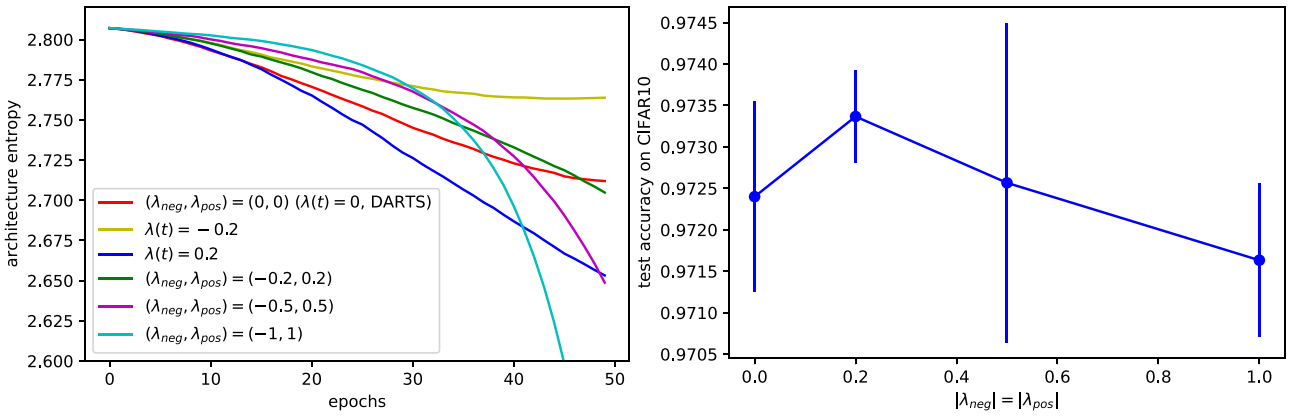
4.4.2. Ablation study for regularizer coefficient scheduling

To find a scheduling way for the regularizer coefficient, we follow the settings in Section 4.4.1 to conduct an ablation study on CIFAR-10. Using a cosine scheduling as a priori method, we explore only the regularizer coefficient pairs $(\lambda_{neg}, \lambda_{pos})$ of starting and ending points. We explore four groups, including $(0.0, 0.0)$, $(-0.2, 0.2)$, $(-0.5, 0.5)$, $(-1.0, 1.0)$. Meanwhile, the other two cases are also considered for comparison, including $\lambda(t) = -0.2$ and $\lambda(t) = 0.2$. We run the search and evaluation three times for each case.

Fig. 6(a) shows that the change process of architecture entropy for all six cases during the search phase. Compared with DARTS, the architecture entropy decreases more slowly in the early stage and faster in the late stage when $(\lambda_{neg}, \lambda_{pos}) = (-0.2, 0.2)$. As shown in Fig. 6(b), when $(\lambda_{neg}, \lambda_{pos}) = (-0.2, 0.2)$, the discovered architectures achieve $97.34\% \pm 0.05$ average test accuracy and significantly outperform others, which proves that our algorithm has better stability and generalization. As we can see in Fig. 6, with $|\lambda_{neg}| = |\lambda_{pos}|$ increasing, the curve of architecture entropy is more extreme and the algorithm tends to degenerate into a worse algorithm whose architecture parameters cannot be learned effectively before the last epoch, making our algorithm more similar to random search.

5. Discussion

Some existing solutions (e.g., early stopping Chu, Zhou et al., 2020; Liang et al., 2019, restricting the number of skip connections Chen et al., 2019; Chu, Zhou et al., 2020; Liang et al., 2019, etc.) cannot help architecture parameters of differentiable NAS escape from local optimization. Our research shows that the



(a) Architecture entropy. (b) The average and standard deviation of test accuracy.

Fig. 6. Ablation study for different scheduling ways of the architecture entropy regularizer coefficient. We explore the case of cosine scheduling with the equal absolute values of λ_{neg} and λ_{opt}. We run the search and evaluation three times on CIFAR-10 for each case. We report the changes in architecture entropies of them during the search phase and the average and standard deviation of the test accuracies of their discovered architectures. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

core of improving differentiable NAS algorithms is how to guide architecture parameters to be learned correctly in the dilemma. In particular, FairNAS (Chu et al., 2021) and DAAS (Tian, Liu, Xie, Jiao, & Ye, 2021) are two extremes of the dilemma and push the search process towards discretized search or not. But our work can provide a smooth transition between the two extremes. We expect that the main improving direction of differentiable NAS algorithms is to explore how to schedule the coefficients of the architecture entropy regularizer to guide architecture parameters to be learned better. It has a promising prospect to find a better way of learning architecture parameters to the optimal, e.g. a heuristic regularizer or an advanced update policy for architecture parameters.

Furthermore, except for the differentiable algorithm itself, there are two major issues in differentiable NAS. (1) The existing differentiable search space is outdated. We cannot expect to observe significant accuracy gain under the current experimental settings in the existing differentiable NAS works. Some works criticize that even the random search with some heuristic tricks (such as setting a fixed number of particular operators or limiting the depth of the cell) can derive competitive performances. In other words, it is hard to judge whether the improvement comes from the search algorithm itself or some random factors. (2) The existing evaluation strategy is unreasonable. Since differentiable NAS usually requires differentiable performance metrics, most of the existing differentiable NAS work use the cross-entropy loss on the validation set for architecture evaluation. However, the cross-entropy loss is a dataset-related metric. This can lead to inconsistent architecture evaluation biases on different datasets. Therefore, the search space with larger variance, more reasonable evaluation strategy, and fairer NAS benchmark are the crucial and urgent demands for NAS, especially differentiable NAS, which can bring vitality to the NAS community.

6. Conclusion

In this work, we propose an architecture entropy as a regularizer for the architecture parameters of differentiable NAS algorithms. By the regularizer coefficient scheduling, we successfully stop DARTS from the dilemma that the existing solutions to the problems of the Matthew effect and discretization discrepancy are inconsistent. We report the best and average error of the architectures we discover, and achieve competitive results, i.e., higher accuracy and better robustness, on CIFAR-10 and ImageNet. Experimental results demonstrate that our architecture entropy regularizer significantly improves different differentiable NAS algorithms on different datasets and different search spaces.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The authors do not have permission to share data.

References

Cai, H., Zhu, L., & Han, S. (2019). ProxylessNAS: Direct neural architecture search on target task and hardware. In *Proc. ICLR'19*.

Chen, X., & Hsieh, C. -J. (2020). Stabilizing differentiable architecture search via perturbation-based regularization. In *Proc. ICML'20* (pp. 1554–1565).

Chen, X., Xie, L., Wu, J., & Tian, Q. (2019). Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *Proc. ICCV'19* (pp. 1294–1303).

Chrabaszcz, P., Loshchilov, I., & Hutter, F. (2017). A downsampled variant of ImageNet as an alternative to the CIFAR datasets. CoRR arXiv:1707.08819.

Chu, X., Zhang, B., & Li, X. (2020). Noisy differentiable architecture search. CoRR arXiv:2005.03566.

Chu, X., Zhang, B., & Xu, R. (2021). FairNAS: Rethinking evaluation fairness of weight sharing neural architecture search. In *Proc. ICCV'21* (pp. 12219–12228).

Chu, X., Zhou, T., Zhang, B., & Li, J. (2020). Fair DARTS: Eliminating unfair advantages in differentiable architecture search. In *Proc. ECCV'20* (pp. 465–480).

Deng, J., Dong, W., Socher, R., Li, L. -J., Li, K., & Li, F. -F. (2009). ImageNet: A large-scale hierarchical image database. In *Proc. CVPR'09* (pp. 248–255).

Dong, X., & Yang, Y. (2019). Searching for a robust neural architecture in four GPU hours. In *Proc. CVPR'19* (pp. 1761–1770).

Dong, X., & Yang, Y. (2020). NAS-Bench-201: Extending the scope of reproducible neural architecture search. In *Proc. ICLR 2020*.

Ferianc, M., Fan, H., & Rodrigues, M. (2020). VINNAS: Variational inference-based neural network architecture search. CoRR arXiv:2007.06103.

Gao, Y., Bai, H., Jie, Z., Ma, J., Jia, K., & Liu, W. (2020). MTL-NAS: Task-agnostic neural architecture search towards general-purpose multi-task learning. In *Proc. CVPR 2020* (pp. 11540–11549). IEEE.

Green, S., Vineyard, C. M., Helinski, R., & Koç, Ç. K. (2020). RAPDARTS: Resource-aware progressive differentiable architecture search. In *Proc. IJCNN'20* (pp. 1–7).

Hong, W., Li, G., Zhang, W., Tang, R., Wang, Y., Li, Z., et al. (2020). DropNAS: Grouped operation dropout for differentiable architecture search. In *Proc. IJCAI'20* (pp. 2326–2332).

Huang, G., Liu, Z., van der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proc. CVPR'17* (pp. 2261–2269).

Krizhevsky, A., & Hinton, G. (2009). *Learning multiple layers of features from tiny images*: *Tech. rep.*, Citeseer.

Li, G., Qian, G., Delgadillo, I. C., Müller, M., Thabet, A. K., & Ghanem, B. (2020). SGAS: Sequential greedy architecture search. In *Proc. CVPR'20* (pp. 1617–1627).

Li, G., Zhang, X., Wang, Z., Li, Z., & Zhang, T. (2019). StacNAS: Towards stable and consistent optimization for differentiable neural architecture search. CoRR arXiv:1909.11926.

Liang, H., Zhang, S., Sun, J., He, X., Huang, W., Zhuang, K., et al. (2019). DARTS+: Improved differentiable architecture search with early stopping. CoRR arXiv:1909.06035.

Liu, H., Simonyan, K., Vinyals, O., Fernando, C., & Kavukcuoglu, K. (2018). Hierarchical representations for efficient architecture search. In *Proc. ICLR'18*.

Liu, H., Simonyan, K., & Yang, Y. (2019). DARTS: Differentiable architecture search. In *Proc. ICLR'19*.

Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L. -J., et al. (2018). Progressive neural architecture search. In *Proc. ECCV'18* (pp. 19–35).

Luo, R., Tian, F., Qin, T., Chen, E., & Liu, T. -Y. (2018). Neural architecture optimization. In *Proc. NeurIPS'18* (pp. 7827–7838).

Ma, N., Zhang, X., Zheng, H. -T., & Sun, J. (2018). ShuffleNet V2: Practical guidelines for efficient CNN architecture design. In *Proc. ECCV'18* (pp. 122–138).

Pham, H., Guan, M. Y., Zoph, B., Le, Q. V., & Dean, J. (2018). Efficient neural architecture search via parameter sharing. In *Proc. ICML'18* (pp. 4092–4101).

Real, E., Aggarwal, A., Huang, Y., & Le, Q. V. (2019). Regularized evolution for image classifier architecture search. In *Proc. AAAI'19* (pp. 4780–4789).

Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y. L., Tan, J., et al. (2017). Large-scale evolution of image classifiers. In *Proc. ICML'17* (pp. 2902–2911).

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., et al. (2015). ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, *115*(3), 211–252.

Sandler, M., Howard, A. G., Zhu, M., Zhmoginov, A., & Chen, L. -C. (2018). MobileNetV2: Inverted residuals and linear bottlenecks. In *Proc. CVPR'18* (pp. 4510–4520).

Stamoulis, D., Ding, R., Wang, D., Lymberopoulos, D., Priyantha, B., Liu, J., et al. (2019). Single-path NAS: Designing hardware-efficient ConvNets in less than 4 hours. In *Proc. ECML PKDD'19* (pp. 481–497).

Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., et al. (2019). MnasNet: Platform-aware neural architecture search for mobile. In *Proc. CVPR'19* (pp. 2820–2828).

Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. In *Proc. ICML'19* (pp. 6105–6114).

Tian, Y., Liu, C., Xie, L., Jiao, J., & Ye, Q. (2021). Discretization-aware architecture search. *Pattern Recognition*, *120*, Article 108186.

Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., et al. (2019). FB-Net: Hardware-aware efficient ConvNet design via differentiable neural architecture search. In *Proc. CVPR'19* (pp. 10734–10742).

Xie, S., Zheng, H., Liu, C., & Lin, L. (2019). SNAS: Stochastic neural architecture search. In *Proc. ICLR'19*.

Xu, Y., Xie, L., Zhang, X., Chen, X., Qi, G. -J., Tian, Q., et al. (2020). PC-DARTS: Partial channel connections for memory-efficient architecture search. In *Proc. ICLR'20*.

Yao, Q., Xu, J., Tu, W. -W., & Zhu, Z. (2020). Efficient neural architecture search via proximal iterations. In *Proc. AAAI'20* (pp. 6664–6671).

Yu, K., Sciuto, C., Jaggi, M., Musat, C., & Salzmann, M. (2020). Evaluating the search phase of neural architecture search. In *Proc. ICLR'20*.

Zela, A., Elsken, T., Saikia, T., Marrakchi, Y., Brox, T., & Hutter, F. (2020). Understanding and robustifying differentiable architecture search. In *Proc. ICLR 2020*.

Zhang, L. L., Yang, Y., Jiang, Y., Zhu, W., & Liu, Y. (2020). Fast hardware-aware neural architecture search. In *Proc. CVPR'20 workshops* (pp. 2959–2967).

Zheng, X., Ji, R., Wang, Q., Ye, Q., Li, Z., Tian, Y., et al. (2020). Rethinking performance estimation in neural architecture search. In *Proc. CVPR'20* (pp. 11353–11362).

Zhou, H., Yang, M., Wang, J., & Pan, W. (2019). BayesNAS: A Bayesian approach for neural architecture search. In *Proc. ICML'19* (pp. 7603–7613).

Zoph, B., & Le, Q. V. (2017). Neural architecture search with reinforcement learning. In *Proc. ICLR'17*.

Zoph, B., Vasudevan, V., Shlens, J., & Le, Q. V. (2018). Learning transferable architectures for scalable image recognition. In *Proc. CVPR'18* (pp. 8697–8710).