β-DARTS: Beta-Decay Regularization for Differentiable Architecture Search

Peng Ye[1]*, Baopu Li[2], Yikang Li[3], Tao Chen[1]†, Jiayuan Fan[1], Wanli Ouyang[4]
[1]Fudan University, [2]BAIDU USA LLC,
[3]Shanghai AI Laboratory, [4]The University of Sydney
yepeng20@fudan.edu.cn

Abstract

Neural Architecture Search (NAS) has attracted increasingly more attention in recent years because of its capability to design deep neural network automatically. Among them, differential NAS approaches such as DARTS, have gained popularity for the search efficiency. However, they suffer from two main issues, the weak robustness to the performance collapse and the poor generalization ability of the searched architectures. To solve these two problems, a simple-but-efficient regularization method, termed as Beta-Decay, is proposed to regularize the DARTS-based NAS searching process. Specifically, Beta-Decay regularization can impose constraints to keep the value and variance of activated architecture parameters from too large. Furthermore, we provide in-depth theoretical analysis on how it works and why it works. Experimental results on NAS-Bench-201 show that our proposed method can help to stabilize the searching process and makes the searched network more transferable across different datasets. In addition, our search scheme shows an outstanding property of being less dependent on training time and data. Comprehensive experiments on a variety of search spaces and datasets validate the effectiveness of the proposed method.

1. Introduction

Neural architecture search (NAS) has attracted lots of interests for its potential to automatize the process of architecture design. Previous reinforcement learning [26, 33] and evolutionary algorithm [25] based methods usually incur massive computation overheads, which hinder their practical applications. To reduce the search cost, a variety of approaches are proposed, including performance estimation [17], network morphisms [2] and one-shot architecture search [13, 21]. In particular, one-shot methods resort to weight sharing technique, which only needs to train a supernet covering all candidate sub-networks once. Based on this weight sharing strategy, differentiable architecture search [21] (namely DARTS, as shown in Fig. 1) relaxes the discrete operation selection problem to learn differentiable architecture parameters, which further improves the search efficiency by alternately optimizing supernet weights and architecture parameters.

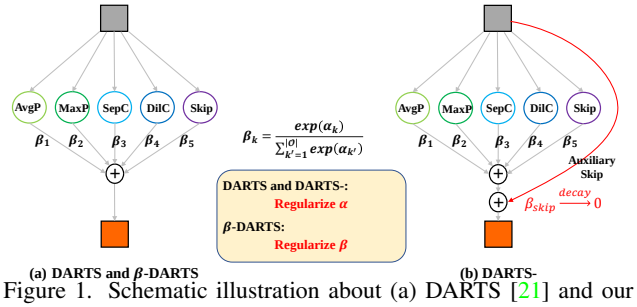


Figure 1. Schematic illustration about (a) DARTS [21] and our proposed β-DARTS, (b) DARTS- [5]. DARTS- adds an auxiliary skip connection with a decay rate β_{skip} to alleviate the performance collapse problem. β-DARTS introduces the Beta-Decay regularization to improve both the robustness of the searching process and the generalization ability of the searched architecture.

Although differentiable method has the advantages of simplicity and computational efficiency, its robustness and architecture generalization challenges still needs to be fully resolved. Firstly, lots of studies have shown that DARTS frequently suffers from performance collapse, that is the searched architecture tends to accumulate parameter-free operations especially for skip connection, leading to the performance degradation [1, 5]. To handle this robustness challenge, lots of instructive works are proposed: directly restricting the number of skip connections [4, 20]; exploiting or regularizing relevant indicators such as the norm of Hessian regarding the architecture parameters [1, 3]; changing the searching and/or discretization process [5, 6, 12]; implicitly regularizing the learned architecture parameters [1]. However, the explicit regularization of architecture parameters optimization receives little attention, as previous works (including above methods) adopt L2 or

*part of this work was done when Ye Peng was tele-interned at Baidu.
†Corresponding author

weight decay regularization by default on learnable architecture parameters (i.e., α), without exploring solution along this direction. Secondly, several works have pointed out that the optimal architecture obtained on the specific dataset cannot guarantee its good performance on another dataset [19,22], namely the architecture generalization challenge. To improve the generalization of searched model, AdaptNAS [19] explicitly minimizes the generalization gap of architectures between domains via the idea of cross domain, MixSearch [22] searches a generalizable architecture by mixing multiple datasets of different domains and tasks. However, both methods solve this issue by leveraging larger datasets, while how to use a single dataset to learn a generalized architecture remains challenging.

This paper is dedicated to simultaneously solve the above-mentioned two challenges in an efficient way. Inspired by the widely-used L2 [7] or weight decay regularization [18] approaches, we intend to design a customized regularization for DARTS-based methods, which can explicitly regularize the optimizing process of architecture parameters. However, different from the regularization on the learnable architecture parameter set, α (before the nonlinear activation of softmax), commonly used in standard DARTS and its subsequent variants, we propose a novel and generic Beta-Decay regularization, imposing regularization on the activated architecture parameters β (after softmax), where $\beta_k = \frac{\exp(\alpha_k)}{\sum_{k'=1}^{|\mathcal{O}|}\exp(\alpha_{k'})}$. On one hand, the proposed Beta-Decay regularization is very simple to implement, achieved with only additional one line of PyTorch code in DARTS (Alg 1). On the other hand, this simple implementation is grounded by in-depth theoretical support. We provide theoretical analysis to show that, Beta-decay regularization not only mitigates unfair competition advantage among operations and solve the domination problem of parameter-free operations, but also minimizes the Lipschitz constraint defined by architecture parameters and make sure the generalization ability of searched architecture. In addition, we mathematically and experimentally demonstrate that, commonly-used L2 or weight decay regularization on α may not be effective or even counterproductive for improving robustness and generalization of DARTS.

Algorithm 1 PyTorch Implementation in DARTS

```
L_Beta = torch.mean(torch.logsumexp(
    self.model._arch_parameters, dim=-1))
loss = self._val_loss(self.model, input_valid,
    target_valid)+λL_Beta
```

DARTS with Beta-Decay regularization (β-DARTS) is illustrated in Fig. 1. Extensive experiments on various search spaces (i.e. NAS-Bench-201, DARTS, NAS-Bench-1Shot1) and datasets (i.e. CIFAR-10, CIFAR-100, ImageNet) verify the effectiveness of our method. Besides, our search scheme shows the following outstanding properties:

- The search trajectories on NAS-Bench-201 and NAS-Bench-1Shot1 show that, the found architecture has continuously rising performance, and the search process can reach its optimal point at an early epoch.
- We only need to search once on the proxy dataset (i.e., CIFAR-10), but the searched architecture can obtain promising performance on various datasets (i.e., CIFAR-10, CIFAR-100 and ImageNet).

2. Related Works

2.1. Robustness of DARTS

As DARTS is known to chronically suffer from the performance collapse issue caused by the domination of parameter-free operators, lots of works have dedicated to resolving it. P-DARTS [4] and DARTS+ [20] directly limit the number of skip connections. Such handcrafted rules are somewhat suspicious and may mistakenly reject good networks. R-DARTS [1] finds that the Hessian eigenvalues can be regarded as an indicator for the collapse, and employs stronger regularization or augmentation on the training of supernet weights to reduce this value. Then SDARTS [3] implicitly regularizes this indicator by adding perturbations to architecture parameters via random smoothing or adversarial attack. Both methods are indirect solutions and rely heavily on the quality of the indicator. FairDARTS [6] avoids operation competition by weighting each operation via independent sigmoid function, which will be pushed to zero or one by an MSE loss. DropNAS [15] proposes a grouped operation dropout for the co-adaption problem and matthew effect. DOTS [12] further uses the group operation search scheme to decouple the operation and topology search. DARTS- [5] factors out the optimization advantage of skip connection by adding an auxiliary one. However, these methods circumvent the domination effect of parameter-free operations by changing the searching and/or discretization process or adding extra parameters. Different from these works, we explore a more generic solution by explicitly regularizing the architecture parameters optimization, making original DARTS great again.

2.2. Generalization of DARTS

Improving the generalization ability of deep model has always been the focus of deep learning research. Recent works provide guarantee on model generalization by minimizing loss value and loss sharpness simultaneously [11]. However, the model generalization is not only related to the network weights, but also determined by its architecture. To this end, several methods attempt to improve the generalization of architectures in the field of NAS. AdaptNAS [19] incorporates the idea of domain adaptation into the search process of DARTS, which can minimize the generalization gap of neural architectures between domains.

MixSearch [22] uses a composited multi-domain multi-task dataset to search a generalizable architecture in a differentiable manner. On one hand, both above methods are built on the assumption of having multiple datasets, while our method is not built on multiple datasets. On the other hand, our focus is on regularizing architecture parameters, which is not investigated in AdaptNAS and MixSearch.

3. Proposed method

3.1. Formulation of DARTS

Following [34], DARTS searches the structure of normal cell and reduction cell to stack the full network. Typically, a cell is defined as a directed acyclic graph (DAG) with N nodes, where each node denotes a latent representation and the information between every two nodes is transformed by an edge. Each edge (i, j) contains several candidate operations, and DARTS applies continuous relaxation via the learnable architecture parameters set α to mix the outputs of different operations, converting the discrete operation selection into a differentiable parameter optimization problem,

$$\bar{O}^{(i,j)}(x) = \sum_{k=1}^{|\mathcal{O}|} \beta_k^{(i,j)} O_k(x), \quad \beta_k^{(i,j)} = \frac{\exp\left(\alpha_k^{(i,j)}\right)}{\sum_{k'=1}^{|\mathcal{O}|} \exp\left(\alpha_{k'}^{(i,j)}\right)} \tag{1}$$

where x and $\bar{O}$ are the input and mixed output of an edge, $\mathcal{O}$ is the candidate operation set, and β denotes the softmax-activated architecture parameter set. In this way, we can perform architecture search in a differentiable manner by solving following bi-level optimization objective,

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

In practice, architecture parameters α and network weights w are alternately updated on the validation and training datasets via gradient descent, and w^* is approximated by one-step forward or current w [21].

3.2. Commonly-used Regularization

In this paper, we intend to improve the robustness and architecture generalization of DARTS by explicitly regularizing the optimizing process of architecture parameters. Thus, we begin with the default settings of previous methods, namely L2 or weight decay regularization on architecture parameters, α. For convenience of analysing, we consider the single-step update of the architecture parameters,

$$\alpha_k^{t+1} \leftarrow \alpha_k^t - \eta_\alpha \cdot \nabla_{\alpha_k} \mathcal{L}_{val} \tag{3}$$

where η_α and $\mathcal{L}_{val}$ are the learning rate of architecture parameters and the corresponding loss respectively. For multi-step updates, it can be transformed into a single-step update problem through step-wise recursive analysis.

In standard DARTS and its subsequent variants, Adam optimizer with L2 regularization is commonly used for architecture parameters optimization. However, for adaptive gradient algorithms, the gradients of L2 regularization are normalized ($\mathcal{N}$) by their summed magnitudes, thus the penalty for each element is relatively even, which partly offsets the effect of L2 regularization [23], defined as

$$\bar{\alpha}_k^{t+1} \leftarrow \alpha_k^t - \eta_\alpha \cdot \nabla_{\alpha_k} \mathcal{L}_{val} - \eta_\alpha \lambda \mathcal{N}\left(\alpha_k^t\right) \tag{4}$$

Considering that L2 regularization may not be effective in adaptive gradient algorithms and is not identical to weight decay regularization [23], without loss of generality, we also include architecture parameters optimization with weight decay regularization [14] for comparison, defined as

$$\bar{\alpha}_k^{t+1} \leftarrow \alpha_k^t - \eta_\alpha \cdot \nabla_{\alpha_k} \mathcal{L}_{val} - \eta_\alpha \lambda \alpha_k^t \tag{5}$$

3.3. Beta-Decay Regularization

Since the searching and discretization process of DARTS actually utilize softmax-activated architecture parameter set, β, to represent the importance of each operator, we shall pay more attention to the explicit regularization on β. As shown in Subsection 3.4, Beta regularization has the ability to improve the robustness and architecture generalization of DARTS, which further denotes its significance. Although important, Beta regularization is typically ignored by previous works. This paper is devoted to filling this gap. Similar to the idea of most regularization methods, the core purpose of Beta regularization is to constrain the value of Beta from changing too much, formulated as

$$\bar{\beta}_k^{t+1} = \theta_k^{t+1}\left(\alpha_k^t\right) \beta_k^{t+1} \tag{6}$$

For simplicity, we use a θ function with α as the independent variable to express the total influence of Beta regularization here. To realize above Beta regularization similar to weight decay through α, we firstly study the influence of α regularization on β. Recalling Eq. (4) and Eq. (5), we can conclude a unified formula as:

$$\bar{\alpha}_k^{t+1} \leftarrow \alpha_k^t - \eta_\alpha \nabla_{\alpha_k} \mathcal{L}_{val} - \eta_\alpha \lambda F\left(\alpha_k^t\right) \tag{7}$$

Further, we substitute Eq. (7) and Eq. (3) into Eq. (1) to get $\bar{\beta}_k^{t+1}$ and β_k^{t+1}, and then divide the former by the latter.

$$\frac{\bar{\beta}_k^{t+1}}{\beta_k^{t+1}} = \frac{\sum_{k'=1}^{|\mathcal{O}|} \exp\left(\alpha_{k'}^{t+1}\right)}{\sum_{k'=1}^{|\mathcal{O}|} \left[\exp\left(F(\alpha_k^t) - F(\alpha_{k'}^t)\right)\right]^{\lambda\eta_\alpha} \exp\left(\alpha_{k'}^{t+1}\right)} \tag{8}$$

As we can see in Eq. (8), the mapping function F determines the influence of α on β. Thus, all we need is to look for a suitable mapping function, F. Intuitively, a satisfactory F should meet the following two points: (1) F is

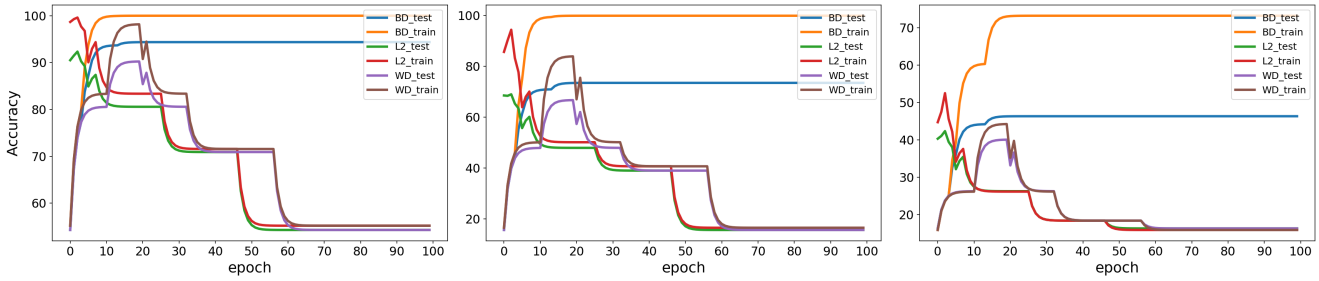


Figure 2. Accuracy of different datasets of DARTS with L2, Weight Decay (WD) and Beta-Decay (BD) regularization on NAS-Bench-201 benchmark. The curve is smoothed with a coefficient of 0.5. Note that we only search once on CIFAR-10 dataset.

not affected by the amplitude of α (to avoid invalid regularization and optimization difficulties). (2) F can reflect the relative amplitude of α (to impose more penalty on larger amplitude). To satisfy the two requirements, we adopt the softmax to normalize α,

$$F\left(\alpha_k\right)=\frac{\exp\left(\alpha_k\right)}{\sum_{k'=1}^{|\mathcal{O}|}\exp\left(\alpha_{k'}\right)} \tag{9}$$

Then we introduce our proposed Beta-Decay regularization loss, whose gradients with respective to α equals to $F\left(\alpha\right)$,

$$\mathcal{L}_{Beta}=\log\left(\sum_{k=1}^{|\mathcal{O}|}e^{\alpha_k}\right)=\text{smoothmax}\left(\{\alpha_k\}\right) \tag{10}$$

After that, substituting Eq. (9) into Eq. (8), we can obtain following equation, which further accounts for the effect of Beta-Decay regularization,

$$\theta_k^{t+1}\left(\alpha_k^t\right)=\frac{\sum_{k'=1}^{|\mathcal{O}|}\exp\left(\alpha_{k'}^{t+1}\right)}{\sum_{k'=1}^{|\mathcal{O}|}\left[\exp\left(\frac{\exp(\alpha_k^t)-\exp(\alpha_{k'}^t)}{\sum_{k''=1}^{|\mathcal{O}|}\exp\left(\alpha_{k''}^t\right)}\right)\right]^{\lambda\eta_\alpha}\exp\left(\alpha_{k'}^{t+1}\right)} \tag{11}$$

Observing the above formula, we can get following conclusions: (1) When α is the largest, θ is the smallest and less than 1; when α is the smallest, θ is the largest and greater than 1; and when α is equal, $\theta=1$. (2) In current iteration, θ decreases as α increases. (3) θ is smaller when α is larger, and θ is larger when α is smaller. As a result, the variance of β is constrained to be smaller, and the value of β is constrained to be closer to its mean, achieving the effect similar to weight decay, thus called Beta-Decay regularization.

3.4. Theoretical Analysis

Stronger Robustness. According to the theorem revealed by recent work [32], the convergence of network weights w can heavily rely on β_{skip} in the supernet. In details, supposing that there are three operations (convolution, skip connection and none) in the search space and the training loss is MSE, when fixing architecture parameters to optimize network weights via gradient descent, at one step the training loss can be reduced by ratio $(1-\eta_w\varphi/4)$ with a probability of at least $1-\sigma$,where η_w is the corresponding learning rate and will be bounded by σ, and φ obeys

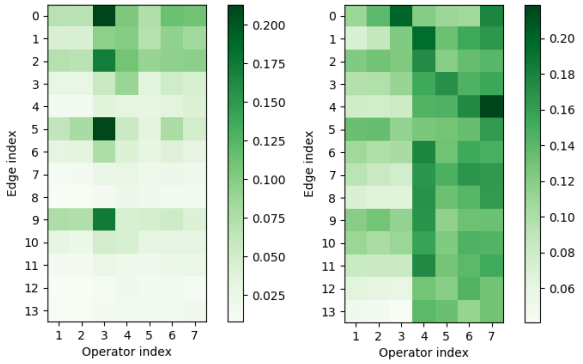


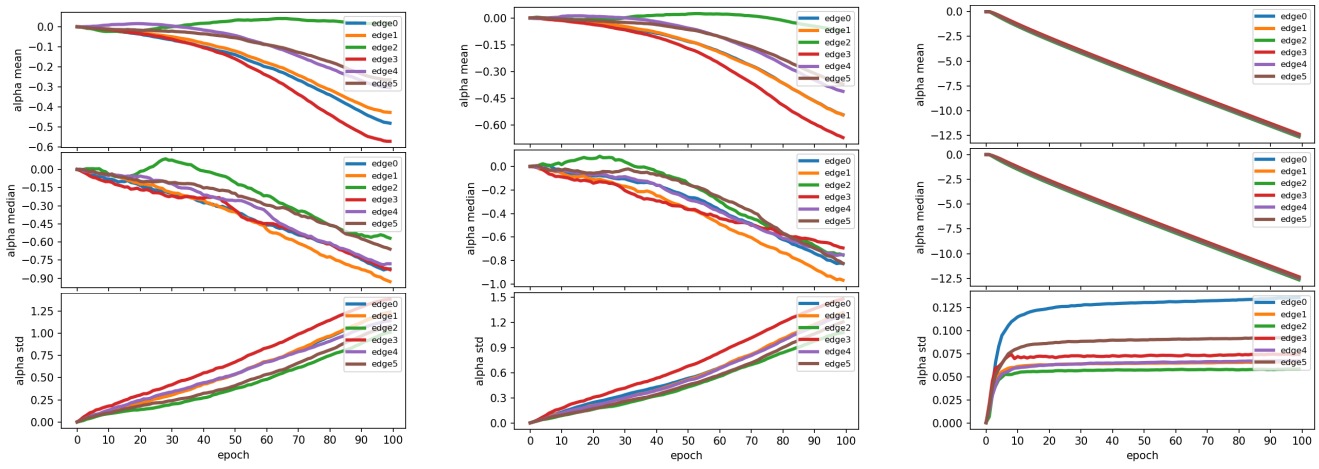
Figure 3. The beta distribution of normal cell learned by DARTS and β-DARTS, on the original search space in CIFAR-10. The operator indexes 1/2/3 mean the max pool/avg pool/skip connect, while others are the parametric operators. The total std is calculated by the sum of the standard deviation of all edges under the edge independence assumption.

$$\varphi\propto\sum_{i=0}^{h-2}\left[\left(\beta_{conv}^{(i,h-1)}\right)^2\prod_{t=0}^{i-1}\left(\beta_{skip}^{(t,i)}\right)^2\right] \tag{12}$$

where h is the number of supernet layers. From Eq. (12), we can see that φ depends more on β_{skip} than β_{conv}, which demonstrates that the supernet weights can converge much faster with large β_{skip}. However, by imposing Beta-Decay regularization, we can redefine Eq. (12) as follows

$$\varphi\propto\sum_{i=0}^{h-2}\left[\left(\theta_{conv}^{(i,h-1)}\beta_{conv}^{(i,h-1)}\right)^2\prod_{t=0}^{i-1}\left(\theta_{skip}^{(i,h-1)}\beta_{skip}^{(t,i)}\right)^2\right] \tag{13}$$

As mentioned before, θ becomes smaller when β is larger and θ becomes larger when β is smaller, which makes the convergence of network weights rely more on β_{conv} and less on β_{skip}. *From the perspective of convergence theorem* [32], *the Beta-Decay regularization constrains the privilege of β_{skip} and ensures the fair competition among architecture parameters*. As shown in Fig. 2, DARTS with L2 or



(a) L2 regularization **(b) weight decay regularization** **(c) Beta-Decay regularization**

Figure 4. The alpha statistical characteristics (i.e. mean, median and standard deviation) of different edges of each epoch when searching on NAS-Bench-201 benchmark with (a) L2 regularization, (b) weight decay regularization, and (c) Beta-Decay regularization.

weight decay regularization suffers from the performance collapse issue, while DARTS with Beta-Decay regularization has a stable search process. As shown in Fig. 3, original DARTS is dominated by skip connections while β-DARTS tends to favor parametric operators.

Stronger Generalization. Referring to [24] and [10], Lipschitz constraint is commonly used to measure and improve the generalization ability of trained deep model. Specifically, suppose the function fitted by a deep model is $f_w(x)$ where x is the input, when $\|x_1 - x_2\|$ is very small, a well-trained model should meet the following constraint.

$$\|f_w(x_1) - f_w(x_2)\| \leq C(w) \cdot \|x_1 - x_2\| \tag{14}$$

where $C(w)$ is the Lipschitz constant. The smaller the constant is, the trained model will be less sensitive to input disturbances and have better generalization ability.

Furthermore, we can extend this theory to differentiable architecture search. For simplicity, we consider a single-layer neural network, and multi-layer neural network can be solved through step-wise recursive analysis. Suppose the single-layer network is mixed by the operation set $F(x) = (f_1(x), f_2(x), f_3(x))$, with the corresponding architecture parameters $\beta = (\beta_1, \beta_2, \beta_3)$. According to the Cauchy's inequality, we can get the following inequality.

$$\left\|\beta F^{\mathrm{T}}(x_1) - \beta F^{\mathrm{T}}(x_2)\right\| \leq \|\beta\| \left\|F^{\mathrm{T}}(x_1) - F^{\mathrm{T}}(x_2)\right\| \tag{15}$$

where $\|\beta\| = \sqrt{\sum \beta_i^2}$ can be regarded as Lipschitz constant and $\sum \beta_i = 1$. *As a result, the smaller the measure $\|\beta\|$ is, the supernet will be less sensitive to the impact of input on the operation set, and the searched architecture will have better generalization ability.* As shown in Fig. 2, the model searched by β-DARTS on CIFAR-10 can well generalize to the CIFAR-100 and ImageNet16 datasets and achieve excellent results. As shown in Fig. 4 and Fig. 3, the architecture parameter distribution learned by β-DARTS maintains a relative small standard deviation, making sure the generalization ability of the searched model.

3.5. Commonly-used Regularization May Not Work

When using L2 regularization on α, we can obtain its effect on β according to Eq. (4) and Eq. (8), defined as

$$\frac{\bar{\beta}_k^{t+1}}{\beta_k^{t+1}} = \frac{\sum_{k'=1}^{|\mathcal{O}|} \exp\left(\alpha_{k'}^{t+1}\right)}{\sum_{k'=1}^{|\mathcal{O}|} \left[\exp\left(\mathcal{N}(\alpha_k^t) - \mathcal{N}(\alpha_{k'}^t)\right)\right]^{\lambda\eta_\alpha} \exp\left(\alpha_{k'}^{t+1}\right)} \tag{16}$$

Similarly, when using weight decay on α, we can obtain its effect on β according to Eq. (5) and Eq. (8), as follows

$$\frac{\bar{\beta}_k^{t+1}}{\beta_k^{t+1}} = \frac{\sum_{k'=1}^{|\mathcal{O}|} \exp\left(\alpha_{k'}^{t+1}\right)}{\sum_{k'=1}^{|\mathcal{O}|} \left[\exp\left(\alpha_k^t - \alpha_{k'}^t\right)\right]^{\lambda\eta_\alpha} \exp\left(\alpha_{k'}^{t+1}\right)} \tag{17}$$

From Eq. (16) and Eq. (17), we can find that: (1) When the values in α are all around 0, achieving the purpose of L2 and weight decay regularization, Alpha regularization has little effect on Beta; while when the values in α are all not near 0, it means that both regularization do not work. (2) For L2 and weight decay regularization on α, only when the median of α is equal to 0, Alpha regularization has the same and correct effect as Beta regularization. (3) A large variance of α is undesirable, which conflicts with the purpose of L2 and weight decay regularization, and makes the optimization process more sensitive to the hyperparameters λ and η_α. In addition, we show the alpha statistical characteristics when searching with different regularization in Fig. 4, we can see that for L2 and weight decay regularization: (1) The mean and median of α continue to decrease and gradually move away from 0. (2) The standard deviation of α

Table 1. Performance comparison on NAS-Bench-201 benchmark [9]. Note that β-DARTS only searches on CIFAR-10 dataset, but can robustly achieve new SOTA on CIFAR-10, CIFAR-100 and ImageNet16-120. Averaged on 4 independent runs of searching.

Methods	Cost (hours)	CIFAR-10		CIFAR-100		ImageNet16-120	
		valid	test	valid	test	valid	test
DARTS(1st) [21]	3.2	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(2nd) [21]	10.2	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 [8]	8.7	89.89±0.08	93.61±0.09	71.34±0.04	70.70±0.30	41.59±1.33	41.71±0.98
SNAS [28]	-	90.10±1.04	92.77±0.83	69.69±2.39	69.34±1.98	42.84±1.79	43.16±2.64
DSNAS [16]	-	89.66±0.29	93.08±0.13	30.87±16.40	31.01±16.38	40.61±0.09	41.07±0.09
PC-DARTS [29]	-	89.96±0.15	93.41±0.30	67.12±0.39	67.48±0.89	40.83±0.08	41.31±0.22
iDARTS [31]	-	89.86±0.60	93.58±0.32	70.57±0.24	70.83±0.48	40.38±0.59	40.89±0.68
DARTS- [5]	3.2	91.03±0.44	93.80±0.40	71.36±1.51	71.53±1.51	44.87±1.46	45.12±0.82
β-DARTS	3.2	**91.55±0.00**	**94.36±0.00**	**73.49±0.00**	**73.51±0.00**	**46.37±0.00**	**46.34±0.00**
optimal	-	91.61	94.37	73.49	73.51	46.77	47.31

increases monotonically. These mathematical and experimental results show that L2 or weight decay regularization commonly used in existing gradient-based methods are not identical to Beta regularization, and may not be effective or even counterproductive. As a comparison, with our proposed Beta-Decay regularization: (1) The mean and median of α are basically equal. (2) When the standard deviation of α increases to a certain extent, it will remain unchanged.

4. Experiments

In this section, we conduct extensive experiments on various search spaces (i.e. NAS-Bench-201, DARTS, NAS-Bench-1Shot1) and datasets (i.e. CIFAR-10, CIFAR-100, ImageNet) to verify the robustness and generalization of β-DARTS, and we further give some experimental insights about DARTS' dependence on training and data. The overall process of β-DARTS is summarized in Alg 2.

Algorithm 2 β-DARTS

Require:
Architecture parameters α; Network weights w; Number of search epochs E; Regularization coefficient adjustment scheme $\lambda_e, e \in \{1, 2, \ldots, E\}$.

1: Construct a supernet and initialize architecture parameters α and supernet weights w
2: For each $e \in [1, E]$ do
3: Update architecture parameters α by descending $\nabla_\alpha \mathcal{L}_{val} + \lambda_e \mathcal{L}_{Beta}$
4: Update network weights w by descending $\nabla_w \mathcal{L}_{train}$
5: Derive the final architecture based on the learned α.

4.1. Results on NAS-Bench-201 Search Space

Settings. NAS-Bench-201 [9] is the most widely used NAS benchmark analyzing various NAS methods. NAS-Bench-201 provides a DARTS-like search space, containing 4 internal nodes with 5 associated operations. The search space consists of 15,625 architectures, with the ground truth performance of CIFAR-10, CIFAR-100 and ImageNet16-120 of each architecture provided. On NAS-Bench-201, the searching settings are kept the same as DARTS on [9].

Results. The comparison results are shown in Table 1. We only search on CIFAR-10 and use the found genotype to query the performance of various datasets. For robustness, our 4 runs of searching under different random seeds always find the same optimal solution, which is very close to the optimal performance of NAS-Bench-201. Moreover, as shown in Fig. 2, the performance collapse issue is well solved and β-DARTS has a stable search process. For generalization ability, we can see that the architecture found on CIFAR-10 achieves consistent new SOTA on CIFAR-10, CIFAR-100 and ImageNet. For dependency on training and data, as shown in Fig. 2, the search process reaches its optimal point at an early stage (i.e., before 20 epochs), on different datasets. Such results validate that β-DARTS has the ability to find the optimal architecture rapidly. More interestingly, we find that the search process of different datasets reach the optimal point in different epochs, although they belong to the same run of searching on CIFAR-10. More similar results can be found in Appendix A.1.

4.2. Results on DARTS Search Space

Settings. Common DARTS search space [21] is also popular for evaluating NAS methods. The search space consists of normal cell and reduction cell. Each cell has 4 intermediate nodes with 14 edges, and each edge is associated with 8 candidate operations. On DARTS search space, all the search settings are kept the same as DARTS since our method only introduces the simple regularization. For evaluation settings, the evaluation on CIFAR-10/100 follows DARTS [21] and the evaluation on ImageNet follows P-DARTS [4] and PC-DARTS [29].

Results. The comparison results are shown in Table 2. We search on CIFAR-10 or CIFAR-100 while evaluating the inferred architecture on CIFAR-10, CIFAR-100 and ImageNet. For robustness, the average results of multiple independent runs of β-DARTS achieve the SOTA performance on both CIFAR-10 and CIFAR-100, namely 97.47±0.08%

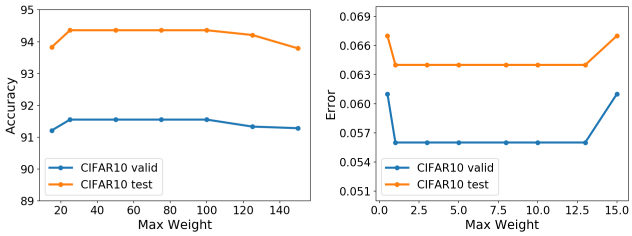
Table 2. Comparison of SOTA models on CIFAR-10/100 (left) and ImageNet(right). For CIFAR-10/100, results in the top block are obtained by training the best searched model while the bottom block shows the average results of multiple runs of searching. ‡ denotes the results of independently searching 3 times on CIFAR-100 and evaluating on both CIFAR-10 and CIFAR-100, while † denotes the results on CIFAR-10. Because of the difference on classifiers, the network parameters on CIFAR-100 is slightly more than that of CIFAR-10 (about 0.05M). For ImageNet, the top block denotes networks are directly searched on ImageNet (Img.), the middle block indicates architectures are searched via the idea of Cross Domain (CD.) using CIFAR-10 and part of ImageNet, models in the bottom block are transferred from the searching results of CIFAR-10 (C10) or CIFAR-100 (C100). * denotes the model is obtained on a different search space.

Method	GPU	CIFAR-10		CIFAR-100	
	(Days)	Params(M)	Acc(%)	Params(M)	Acc(%)
NASNet-A [33]	2000	3.3	97.35	3.3	83.18
DARTS(1st) [21]	0.4	3.4	97.00±0.14	3.4	82.46
DARTS(2nd) [21]	1	3.3	97.24±0.09	-	-
SNAS [28]	1.5	2.8	97.15±0.02	2.8	82.45
GDAS [8]	0.2	3.4	97.07	3.4	81.62
P-DARTS [4]	0.3	3.4	97.50	3.6	82.51
PC-DARTS [29]	0.1	3.6	97.43±0.07	3.6	83.10
P-DARTS [4]	0.3	3.3±0.21	97.19±0.14	-	-
R-DARTS(L2) [1]	1.6	-	97.05±0.21	-	81.99±0.26
SDARTS-ADV [3]	1.3	3.3	97.39±0.02	-	-
DOTS [12]	0.3	3.5	97.51±0.06	4.1	83.52±0.13
DARTS+PT [27]	0.8	3.0	97.39±0.08	-	-
DARTS- [5]	0.4	3.5±0.13	97.41±0.08	3.4	82.49±0.25
β-DARTS‡	0.4	3.78±0.08	97.49±0.07	3.83±0.08	83.48±0.03
β-DARTS†	0.4	3.75±0.15	97.47±0.08	3.80±0.15	83.76±0.22

Method	GPU	Params	FLOPs	Top1	Top5
	(Days)	(M)	(M)	(%)	(%)
MnasNet-92*(Img.) [26]	1667	4.4	388	74.8	92.0
FairDARTS*(Img.) [6]	3	4.3	440	75.6	92.6
PC-DARTS(Img.) [29]	3.8	5.3	597	75.8	92.7
DOTS(Img.) [12]	1.3	5.3	596	76.0	92.8
DARTS-*(Img.) [5]	4.5	4.9	467	76.2	93.0
AdaptNAS-S(CD.) [19]	1.8	5.0	552	74.7	92.2
AdaptNAS-C(CD.) [19]	2.0	5.3	583	75.8	92.6
AmoebaNet-C(C10) [25]	3150	6.4	570	75.7	92.4
SNAS(C10) [28]	1.5	4.3	522	72.7	90.8
P-DARTS(C100) [4]	0.3	5.1	577	75.3	92.5
SDARTS-ADV(C10) [3]	1.3	5.4	594	74.8	92.2
DOTS(C10) [12]	0.3	5.2	581	75.7	92.6
DARTS+PT(C10) [27]	0.8	4.6	-	74.5	92.0
β-DARTS(C100)	0.4	5.4	597	75.8	92.9
β-DARTS(C10)	0.4	5.5	609	76.1	93.0

Table 3. Influence of different weighting schemes on β-DARTS.

Weighting Scheme	CIFAR-10 valid	CIFAR-10 test
0-15/25/50/100	91.21/91.55/91.55/91.55	93.83/94.36/94.36/94.36
5/10/15/25	84.96/90.59/91.55/90.59	88.02/93.31/94.36/93.31
25-15/10/5/0	90.59/87.30/73.58/39.77	93.31/90.65/76.88/54.30



(a) β-DARTS on NAS-Bench-201 **(b) β-SDARTS-RS on NAS-Bench-1Shot1**

Figure 5. The effect of different max weight of linear increased weighting schemes on the searching results.

and 83.48±0.03%, without extra changes or any cost. For generalization ability, architectures found on CIFAR-100 can still yield a SOTA result of 97.49±0.07% on CIFAR-10, and models found on CIFAR-10 obtain a new SOTA of 83.76±0.22% on CIFAR-100, and networks found on both CIFAR-10 and CIFAR-100 datasets can achieve comparable results on ImageNet with those of directly searching on ImageNet or using cross domain method.

4.3. Ablation Study

Importance of Increased Weighting Scheme. We firstly explore the influence of different weighting schemes on β-DARTS, including linear increased weighting scheme, constant weighting scheme and linear decay weighting scheme. The results are shown in Table 3. As we can see, linear decay weighting scheme impedes the effect of regularization, constant weighting scheme is sensitive to the hyperparameter, while linear increased weighting scheme is not only effective but also insensitive to hyperparameter. Besides, combining with the results of Fig. 2 that the performance on CIFAR-10, CIFAR-100, and Imagenet in single run of searching reach the optimal point in order, we conclude that linear increased regularization coefficient can further maximize the generalization ability of inferred model after the searching performance on current data is maximized, as evidenced by Eq. (11) and Eq. (15).

Wide Range of The Optimal Weight. We further investigate the optimal max weight of linear increased weighting scheme. The results on CIFAR-10 of NAS-Bench-201 and search space 1 of NAS-Bench-1Shot1 are provided in Fig. 5. We can see that the best performance is achieved in a wide range of max weights, namely about 25-100 and 1-13 for β-DARTS on NAS-Bench-201 and β-SDARTS-RS on NAS-Bench-1Shot1 respectively. There are similar results on CIFAR-10 and CIFAR-100 in common DARTS search space, as shown in Appendix A.2. If not mentioned specially, the default values of max weight for NAS-Bench-201, NAS-Bench-1Shot1, CIFAR-10 and CIFAR-100 are set to 50, 7, 0.5, 5 respectively in all our experiments. Furthermore, comparing Eq. (11) with Eq. (16) and Eq. (17), we find that the normalized values of α in Eq. (11) has the ability to make sure that the optimization process is not sensitive to the hyperparameter of λ.

4.4. Discussions

Non-uniqueness. Actually, the idea of Beta regularization is what really matters, and the way to realize it is non-unique. Here, we show two kinds of variants of Beta regu-

Table 4. The results of different Beta regularization loss with different weighting schemes on NAS-Bench-201 benchmark. Note that we only search on CIFAR-10 dataset, and perform 2 runs of searching under different random seeds.

Methods	Weighting Scheme	CIFAR-10		CIFAR-100		ImageNet16-120	
		valid	test	valid	test	valid	test
DARTS(1st) [21]	3.2	39.77±0.00	54.30±0.00	15.03±0.00	15.61±0.00	16.43±0.00	16.32±0.00
Beta-Global	0-25	91.55/91.55	94.36/94.36	73.49/73.49	73.51/73.51	46.37/46.37	46.34/46.34
Beta-Global	0-50	91.55/91.55	94.36/94.36	73.49/73.49	73.51/73.51	46.37/46.37	46.34/46.34
Beta-Global	0-75	91.55/91.55	94.36/94.36	73.49/73.49	73.51/73.51	46.37/46.37	46.34/46.34
Beta-Global	0-100	91.21/91.55	93.83/94.36	71.60/73.49	71.88/73.51	45.75/46.37	44.65/46.34
Beta-Zero	0-25	91.21/90.97	93.83/93.91	71.60/70.41	71.88/70.78	45.75/43.77	44.65/44.78
Beta-Zero	0-50	91.55/91.21	94.36/93.83	73.49/71.60	73.51/71.88	46.37/45.74	46.34/44.65
Beta-Zero	0-75	91.61/91.05	94.37/93.66	72.75/71.02	73.22/71.38	45.56/45.23	46.71/44.70
Beta-Zero	0-100	91.21/91.21	93.83/93.83	71.60/71.60	71.88/71.88	45.75/45.75	44.65/44.65

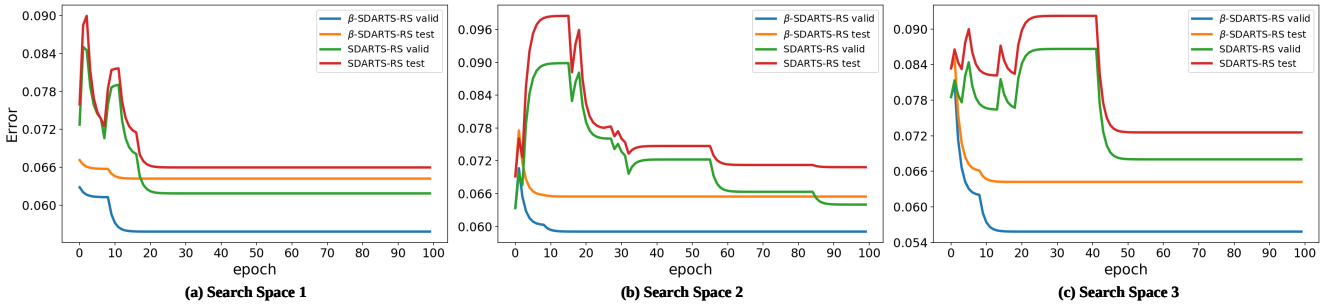


Figure 6. Error of SDARTS-RS and β-SDARTS-RS on 3 search spaces of NAS-Bench-1Shot1 [30]. The curve is smoothed with 0.5.

larization loss. Recalling Eq. (10), we can naturally figure out an alternative, using the $\mathrm{smoothmax}$ of all architecture parameters on the entire supernet, namely Beta-Global loss.

$$\begin{aligned}\mathcal{L}_{Beta-Global} &= \mathrm{smoothmax}\left(\ \alpha_1^1, \cdots, \alpha_{|\mathcal{O}|}^L\ \right) \\ &= \log\left(\sum_{l=1}^{L}\sum_{k=1}^{|\mathcal{O}|} e^{\alpha_k^l}\right)\end{aligned} \tag{18}$$

In addition, by introducing a threshold, we can get the $\mathrm{smoothmax}$ between the threshold and each architecture parameter. We simply set the threshold as 0 in this paper, namely Beta-Zero loss.

$$\begin{aligned}\mathcal{L}_{Beta-Zero} &= \mathrm{smoothmax}\left(\ 0, \alpha_k^l\ \right) \\ &= -\log\left(1 + e^{-\alpha_k^l}\right)\end{aligned} \tag{19}$$

The results of DARTS with Beta-Global and Beta-Zero regularization loss are shown in Table. 4. As we can see, both loss can promote original DARTS by a large margin, while Beta-Global loss that takes the same effect with Beta-Decay loss, can more stably obtain better results than Beta-Zero loss under different weighting schemes. Such results validate that regularizing β is important, while the way to achieve it has a lot of room for exploration.

Generality. Moreover, we utilize NAS-Bench-1Shot1 [30] benchmark and SDARTS-RS [3] baseline to demonstrate the generality of Beta-Decay regularization. NAS-Bench-1Shot1 contains 3 search spaces, which consist of 6,240, 29,160 and 363,648 architectures with the CIFAR-10 performance separately. On NAS-Bench-1Shot1, both the operator of each edge and the topology of the cell need to be determined. We show the search trajectory in Fig. 6. On one hand, β-SDARTS-RS can yield much lower test/validation error than SDARTS-RS across different search spaces. On the other hand, the error of β-SDARTS-RS keeps decreasing while the error of SDARTS-RS increases first and then decreases, validating the more stable search process of β-SDARTS-RS. Besides, the search process of β-SDARTS-RS also reaches its optimal point at an early stage (i.e., around 10 epochs), on different search spaces.

5. Conclusion

In this paper, we investigate the explicit regularization on the optimization of architecture parameters of DARTS in depth, which is typically ignored by previous works. Firstly, we identify that L2 or weight decay regularization on alpha commonly used by DARTS and its variants may not be effective or even counterproductive. Then, we propose a novel and generic Beta-Decay regularization loss, for improving DARTS-based methods without extra changes or cost. In addition, we theoretically and experimentally show Beta-Decay regularization can improve both the robustness and the generalization of DARTS. Besides, we find that the proposed search scheme is less dependent on training time and data. Extensive experiments on various search spaces and datasets validate the superiority of our method.

References

[1] Thomas Elsken Arber Zela, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. Understanding and robustifying differentiable architecture search. In *International Conference on Learning Representations*, volume 3, page 7, 2020. 1, 2, 7

[2] Han Cai, Jiacheng Yang, Weinan Zhang, Song Han, and Yong Yu. Path-level network transformation for efficient architecture search. In *International Conference on Machine Learning*, pages 678–687. PMLR, 2018. 1

[3] Xiangning Chen and Cho-Jui Hsieh. Stabilizing differentiable architecture search via perturbation-based regularization. In *International Conference on Machine Learning*, pages 1554–1565. PMLR, 2020. 1, 2, 7, 8

[4] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1294–1303, 2019. 1, 2, 6, 7

[5] Xiangxiang Chu, Xiaoxing Wang, Bo Zhang, Shun Lu, Xiaolin Wei, and Junchi Yan. Darts-: robustly stepping out of performance collapse without indicators. *arXiv preprint arXiv:2009.01027*, 2020. 1, 2, 6, 7

[6] Xiangxiang Chu, Tianbao Zhou, Bo Zhang, and Jixiang Li. Fair darts: Eliminating unfair advantages in differentiable architecture search. In *European conference on computer vision*, pages 465–480. Springer, 2020. 1, 2, 7

[7] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. L2 regularization for learning kernels. *arXiv preprint arXiv:1205.2653*, 2012. 2

[8] Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1761–1770, 2019. 6, 7

[9] Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. *arXiv preprint arXiv:2001.00326*, 2020. 6

[10] Chris Finlay, Jeff Calder, Bilal Abbasi, and Adam Oberman. Lipschitz regularized deep neural networks generalize and are adversarially robust. *arXiv preprint arXiv:1808.09540*, 2018. 5

[11] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020. 2

[12] Yu-Chao Gu, Li-Juan Wang, Yun Liu, Yi Yang, Yu-Huan Wu, Shao-Ping Lu, and Ming-Ming Cheng. Dots: Decoupling operation and topology in differentiable architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12311–12320, 2021. 1, 2, 7

[13] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *European Conference on Computer Vision*, pages 544–560. Springer, 2020. 1

[14] Stephen Hanson and Lorien Pratt. Comparing biases for minimal network construction with back-propagation. *Advances in neural information processing systems*, 1:177–185, 1988. 3

[15] Weijun Hong, Guilin Li, Weinan Zhang, Ruiming Tang, Yunhe Wang, Zhenguo Li, and Yong Yu. Dropnas: Grouped operation dropout for differentiable architecture search. In *IJCAI*, pages 2326–2332, 2020. 2

[16] Shoukang Hu, Sirui Xie, Hehui Zheng, Chunxiao Liu, Jianping Shi, Xunying Liu, and Dahua Lin. Dsnas: Direct neural architecture search without parameter retraining. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12084–12092, 2020. 6

[17] Aaron Klein, Stefan Falkner, Jost Tobias Springenberg, and Frank Hutter. Learning curve prediction with bayesian neural networks. 2016. 1

[18] Anders Krogh and John A Hertz. A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pages 950–957, 1992. 2

[19] Yanxi Li, Zhaohui Yang, Yunhe Wang, and Chang Xu. Adapting neural architectures between domains. *Advances in Neural Information Processing Systems*, 33, 2020. 2, 7

[20] Hanwen Liang, Shifeng Zhang, Jiacheng Sun, Xingqiu He, Weiran Huang, Kechen Zhuang, and Zhenguo Li. Darts+: Improved differentiable architecture search with early stopping. *arXiv preprint arXiv:1909.06035*, 2019. 1, 2

[21] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018. 1, 3, 6, 7, 8

[22] Luyan Liu, Zhiwei Wen, Songwei Liu, Hong-Yu Zhou, Hongwei Zhu, Weicheng Xie, Linlin Shen, Kai Ma, and Yefeng Zheng. Mixsearch: Searching for domain generalized medical image segmentation architectures. *arXiv preprint arXiv:2102.13280*, 2021. 2, 3

[23] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 3

[24] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nathan Srebro. Exploring generalization in deep learning. *arXiv preprint arXiv:1706.08947*, 2017. 5

[25] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789, 2019. 1, 7

[26] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019. 1, 7

[27] Ruochen Wang, Minhao Cheng, Xiangning Chen, Xiaocheng Tang, and Cho-Jui Hsieh. Rethinking architecture selection in differentiable nas. *arXiv preprint arXiv:2108.04392*, 2021. 7

[28] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926*, 2018. 6, 7

[29] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. Pc-darts: Partial channel connections for memory-efficient architecture search. *arXiv preprint arXiv:1907.05737*, 2019. 6, 7

[30] Arber Zela, Julien Siems, and Frank Hutter. Nas-bench-1shot1: Benchmarking and dissecting one-shot neural architecture search, 2020. 8

[31] Miao Zhang, Steven Su, Shirui Pan, Xiaojun Chang, Ehsan Abbasnejad, and Reza Haffari. idarts: Differentiable architecture search with stochastic implicit gradients. *arXiv preprint arXiv:2106.10784*, 2021. 6

[32] Pan Zhou, Caiming Xiong, Richard Socher, and Steven CH Hoi. Theory-inspired path-regularized differential network architecture search. *arXiv preprint arXiv:2006.16537*, 2020. 4

[33] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018. 1, 7

[34] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018. 3