Differentiable Architecture Search with Random Features

Xuanyang Zhang*[1] Yonggang Li*[2] Xiangyu Zhang[1] Yongtao Wang[2] Jian Sun[1]
[1]MEGVII Technology, [2]Peking University
xuanyang91.zhang@gmail.com {zhangxiangyu, sunjian}@megvii.com
{liyonggang, wangyongtao}@pku.edu.cn

Abstract

*Differentiable architecture search (DARTS) has significantly promoted the development of NAS techniques because of its high search efficiency and effectiveness but suffers from performance collapse. In this paper, we make efforts to alleviate the performance collapse problem for DARTS from two aspects. First, we investigate the expressive power of the supernet in DARTS and then derive a new setup of DARTS paradigm with only training BatchNorm. Second, we theoretically find that random features dilute the auxiliary connection role of skip-connection in supernet optimization and enable search algorithm focus on fairer operation selection, thereby solving the performance collapse problem. We instantiate DARTS and PC-DARTS with random features to build an improved version for each named RF-DARTS and RF-PCDARTS respectively. Experimental results show that RF-DARTS obtains **94.36%** test accuracy on CIFAR-10 (which is the nearest optimal result in NAS-Bench-201), and achieves the newest state-of-the-art top-1 test error of **24.0%** on ImageNet when transferring from CIFAR-10. Moreover, RF-DARTS performs robustly across three datasets (CIFAR-10, CIFAR-100, and SVHN) and four search spaces (S1-S4). Besides, RF-PCDARTS achieves even better results on ImageNet, that is, **23.9%** top-1 and **7.1%** top-5 test error, surpassing representative methods like single-path, training-free, and partial-channel paradigms directly searched on ImageNet.*

1. Introduction

Differentiable architecture search (DARTS) [27] has demonstrated both higher search efficiency and better search efficacy than early pioneering neural architecture search (NAS) [1,50,51] attempts in the image classification task. In the past few years, many following works further improve DARTS by introducing additional modules, such as Gumbel-softmax [13], early stop criterion [23], auxiliary skip-connection [10], etc. We have witnessed tremendous improvements in the image recognition task, but it is getting farther away from exploring how DARTS works. Newly, in this work, we intend to demystify DARTS by disassembling key modules rather than make it more complex.

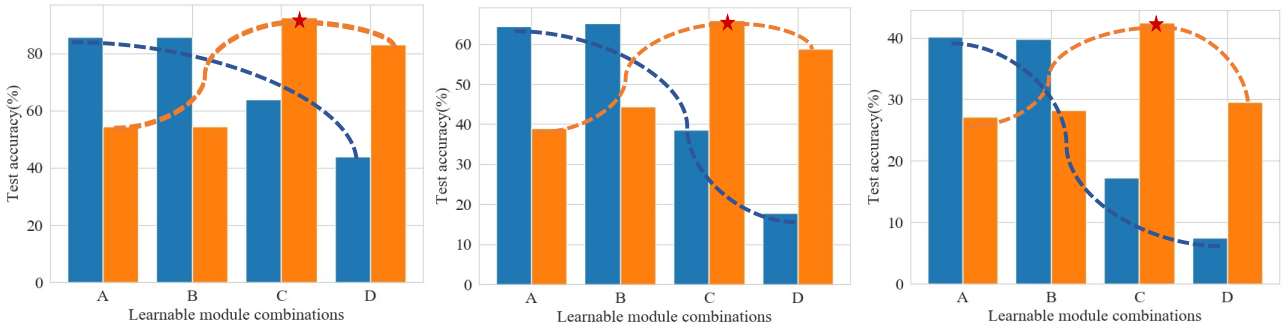
We overview the vanilla DARTS paradigm, and summary three key modules, namely *dataset*, *evaluation metric*, and *supernet* as follows:

- ***Dataset.*** DARTS [27] searches on proxy dataset and then transfers to target dataset due to huge requirements for GPU memory. PC-DARTS [36] proves that proxy datasets inhibit the effectiveness of DARTS and directly searching on target dataset obtains more promising architectures. UnNAS [25] and RLNAS [46] ablate the role of labels in DARTS, and further conclude that ground truth labels are not necessary for DARTS.

- ***Evaluation metric.*** DARTS [27] introduces architecture parameters to reflect the strengths of the candidate operations. PT-DARTS [32] suspects the effectiveness of architecture parameters and shows that the magnitude of architecture parameters does not necessarily indicate how much the operation contributes to the supernet's performance. FreeNAS [45] and TE-NAS [7] further put forward training-free evaluation metrics to predict the performance of candidate architectures.

- ***Supernet.*** DARTS encodes all candidate architectures in search space into the supernet. The search cell of supernet will change as the search space changes. R-DARTS [41] proposes four challenging search spaces S1-S4 where DARTS obtains inferior performance than the Random-search baseline. R-DARTS attributes the failure of vanilla DARTS to the dominant skip-connections. Thus R-DARTS concludes that the topology of supernet has great influence on the efficacy of DARTS. P-DARTS [9] finds that the depth gap

*Equal contributions.This work is done during Yonggang Li's internship at MEGVII Technology. This work is supported by Science and Technology Innovation 2030-New Generation Artificial Intelligence (2020AAA0104401).



(a) Results on CIFAR-10 (b) Results on CIFAR-100 (c) Results on ImageNet16-120

Figure 1. The correlation between the DARTS supernet performance (blue histograms) and the searched architecture performance (orange histograms) in NAS-Bench-201 [14] search space (best viewed in color). We directly searched on target datasets CIFAR-10, CIFAR-100, and ImageNet16-120 in three runs and plot average results. Histograms on three datasets reflect a consistent phenomenon. Supernet performance and the expressive power are positively correlated. However, supernet with higher performance can not search for better architectures, and supernet optimized with only BN has the best search effect.

between search and evaluation architecture prevents DARTS from achieving better search results.

Option (**state**)	Learnable modules in supernet		Expressive Power
	Convolution	BN Affine	
A (**unexplored**)	✓	✓	strong
B (**explored**)	✓	✗	⇓
C (**unexplored**)	✗	✓	⇓
D (**unexplored**)	✗	✗	weak

Table 1. Enumerate four combinations of learnable modules in supernet. We keep the composition of supernet unchanged and just change the optimizable weights. ✓means updating weights with the optimizer. ✗means freezing weights at the initialization.

In this paper, we take a further step to investigate the supernet in DARTS from two respects. First, we ablate the expressive power of supernet in DARTS. Specifically, each convolution operation (like separable convolutions) consists of two learnable modules: convolution layers and BatchNorm (BN) [21] layers. Each convolution layer is followed by a private BN layer. To study the expressive power in isolation, we thoroughly traverse four combinations (named A, B, C, and D for simple) of learnable modules as shown in Tab. 1. Existing DARTS variants [9, 27, 36] adopt option B by default, which disables BN affine weights and only trains convolution weights during the supernet training. On the contrary, option A, C, and D are still unexplored, thus it is a mystery what will happen when supernet is equipped with different expressive power, that is, trained with option of A, C, and D. Hence, we make a sanity check between supernet performance and searched architecture performance across the above four combinations. As shown in Fig. 1, the relative ranking of supernet performance is A≈B>C>D, which is consistent with the ranking of supernet's expressive power. However, the relative ranking of searched architecture performance is C≫D>A≈B, which sounds counintuitive. This result implies that **the performance of supernet is not such significant and scaling random features with BN affine weights is good enough for architecture search**. Therefore, we propose a new extension of DARTS with random features driven by the surprising results of only training BN. Second, we explore the working mechanism of random features by considering skip-connection roles in DARTS. Skip-connection in DARTS plays two roles [10]: 1) as a shortcut to help the optimization of supernet, and 2) a candidate operation for architecture search. Random features dilute the role of auxiliary connection that skip-connection plays in supernet training and enable DARTS to focus on fairer operation selection, thus implicitly solving performance collapse of DARTS [10,41].

Based on the versatility of supernet optimization, we arm popular DARTS [27] and PC-DARTS [36] with random features to build more effective algorithms RF-DARTS and RF-PCDARTS. On CIFAR-10, RF-DARTS obtains **94.36%** test accuracy that is the nearest optimal results (94.37%) in NAS-Bench-201. RF-DARTS achieves the state-of-the-art **24.0%** top-1 test error on ImageNet when transferred from CIFAR-10 in DARTS search space. RF-DARTS also performs robustly on CIFAR-10, CIFAR-100, and SVHN across S1-S4. RF-PCDARTS directly searches on ImageNet and achieves **23.9%** top-1 test error, which surpasses representative methods from single-path, training-free, and partial channel paradigms. Overall, comprehensive results reveal that the expressive power of DARTS supernet is over-powerful, and random features is just perfect for DARTS. We hope these essential analyses and results will inspire new understandings for NAS.

2. Related Work

Differentiable architecture search. With the promising search efficiency, the variants of differentiable architecture search (DARTS) [27] have achieved remarkable performance improvement in various computer vision tasks, like image classification [31, 33], object detection [35], and image segmentation [24]. However, DARTS achieves an efficient search with the cost of several optimization gaps [4, 9, 13, 23, 34, 36–38, 40, 48] between the search and retraining stage. Nevertheless, except for the above optimization gaps, DARTS suffers from the severe performance collapse [2, 8, 10, 11, 23, 41], which hinders a wide range of applications of DARTS algorithms. DARTS- [10] finds that the skip-connection has the advantage for stabilizing the supernet training, and thereby supernet has the tendentiousness for choosing skip-connection compared with other operations. Therefore, they introduce an auxiliary skip-connection to decouple its role for stabilizing the gradient flow and role as a candidate operation. Compared to it, RF-DARTS provides a straightforward method — directly training supernet with random features, to prevent the performance collapse brought by skip-connection.

Random features. Learning with random features [3, 16, 29, 39], which fixes the neural network's weights at initialization, has been developed a long time and has considerable expressive power [16, 39] for building networks. Recently, Frankle *et al.* [15] investigated the performance achieved by random features when training only the affine parameters of BatchNorm (BN) [21]. Nevertheless, there are few works to explore the expressive power of random features for neural architecture search. Although both BN-NAS and BNNAS++ training only BN in NAS community, RF-DARTS has three main difference from BN-NAS [6] and BNNAS++ [49]. Firstly, the motivations are quite different. Performance collapse in DARTS is an important open problem and a lot of previous work like DARTS+ [23], DARTS- [10]and etc try to alleviates this problem. To the best of our knowledge, it is brand new that training only BN and still using architecture parameter to distinguish operations can solve the performance collapse problem. However, both BN-NAS and BNNAS++ propose a new model evaluation metric to improve the search efficiency. Secondly, BN-NAS and BNNAS++ are applied for SPOS [18] framework while RF-DARTS is for gradient-based NAS. BNNAS and BNNAS++ have to introduce additional BN layers for each operations while it is not necessary for RF-DARTS because of architecture parameters. This property shrinks the gap between supernet and searched architecture, thus RF-DARTS can performance robustly across both simple search space like NAS-Bench-201, complex search space like DARTS and Robust DARTS search space. Thirdly, this paper explores the working mechanism of random features by considering skip-connection roles in DARTS. Random features dilute the role of auxiliary connection that skip-connection plays in supernet training and enable DARTS to focus on fairer operation selection. Besides, both theoretical and empirical analysis of random features from the perspective of gradient in DARTS are also new tools to understand NAS.

3. Methodology

3.1. Preliminary: DARTS

DARTS searches for the shared cells for a network with L layers, where each cell is a directed acyclic graph (DAG) with N nodes. Given the pre-defined candidate operations set $\mathcal{O}$ with candidate operation $o(\cdot)$ (e.g., skip-connection, convolution), DARTS needs to determine the operation selection $o \in \mathcal{O}$ for each edge between every two nodes of the shared cell. Rather than directly making the categorical choice of a certain operation like ENAS [28], DARTS relaxes the search space to the continuous one through a softmax over all operations in the candidate set $\mathcal{O}$:

$$\bar{o}(x, w_{\text{conv}}, \alpha) = \sum_{o \in \mathcal{O}} \frac{\exp(\alpha_o)}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'})} o(x, w^o_{\text{conv}}), \quad (1)$$

where parameter $\alpha \in \mathbb{R}^{|\mathcal{O}|}$ represents the operations mixing weight, and w_{conv} is the weight of convolution layer for each operation (*DARTS disables the affine weights of BatchNorm (BN), thus only optimizes the weights of the convolution layer*). Thereby, for determining a neural architecture with high validation performance, DARTS aims to optimize the architecture parameters α and the convolution weights w_{conv} of supernet jointly, and finally uses the optimized parameters α to derive discrete architectures.

Formally, DARTS seek to find the architecture parameters α^* with minimal validation loss $\mathcal{L}_{\text{val}}(w^*_{\text{conv}}, \alpha^*)$ where the architecture weights w^*_{conv} are optimized by minimizing the training loss $\mathcal{L}_{\text{train}}(w_{\text{conv}}, \alpha^*)$. This objective can be represented as the below *bi-level optimization* formulation:

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

$$\text{s.t.} \quad w^*_{\text{conv}}(\alpha) = \underset{w_{\text{conv}}}{\operatorname{argmin}} \ \mathcal{L}_{\text{train}}(w_{\text{conv}}, \alpha). \quad (3)$$

However, it is tough to solve the above nested optimization problem directly due to the prohibitive gradient of α w.r.t. $\mathcal{L}_{\text{val}}$. Therefore, DARTS proposes to approximate the gradient using only one single training step as below:

$$\nabla_\alpha \mathcal{L}_{\text{val}}(w^*_{\text{conv}}(\alpha), \alpha)) \approx \quad (4)$$

$$\nabla_\alpha \mathcal{L}_{\text{val}}(w_{\text{conv}} - \xi \nabla_{w_{\text{conv}}} \mathcal{L}_{\text{train}}(w_{\text{conv}}, \alpha), \alpha)). \quad (5)$$

In this way, DARTS iteratively optimize the weights w_{conv} and parameters α jointly through gradient descent.

3.2. RF-DARTS: DARTS with Random Features

The supernet of DARTS usually contains two learnable modules – convolution (Conv) and BatchNorm (BN) layer. However, DARTS and its variants only consider optimizing the weights of Conv but freezing the learnable affine weights of BN. To understand the expressive power of the above two learnable modules in the weight-sharing supernet, we extend DARTS by introducing the learnable affine weights of BN w_{bn} to the search stage of supernet:

$$\bar{o}(x, w_{\text{conv}}, w_{\text{bn}}, \alpha) = \sum_{o \in \mathcal{O}} \frac{\exp(\alpha_o)}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'})} o(x, w^o_{\text{conv}}, w^o_{\text{bn}}). \tag{6}$$

Through the analyses of Sec. 4, we find that it is advantageous for search promising architecture when we optimize the weights of the Conv and BN in an inverse way as DARTS. Consequently, we propose the DARTS with random features (RF-DARTS), which only updates the weights of BN but freezes the weights of the Conv at initialization. Hence, the optimization object of RF-DARTS can be formulated as follows:

$$\min_{\alpha} \quad \mathcal{L}_{\text{val}}(w^{\text{init}}_{\text{conv}}, w^*_{\text{bn}}(\alpha), \alpha), \tag{7}$$

$$\text{s.t.} \quad w^*_{\text{bn}}(\alpha) = \operatorname*{argmin}_{w_{\text{bn}}} \mathcal{L}_{\text{train}}(w^{\text{init}}_{\text{conv}}, w_{\text{bn}}, \alpha), \tag{8}$$

where $w^{\text{init}}_{\text{conv}}$ is kept at initialization and only w_{bn} and α are optimized. To solve the above objective, we alternatively optimize the architecture parameters α and the weights of BN w_{bn} in a similar way as DARTS. Furthermore, to accelerate the search procedure, we simply utilize the first-order approximation of gradient by setting $\xi = 0$ when calculating gradient $\nabla_\alpha \mathcal{L}_{\text{val}}(w_{\text{conv}}, w_{\text{bn}} - \xi \nabla_{w_{\text{bn}}}, \alpha)$.

4. Analysis and Discussions

As verified by Frankle *et al.* [15], the random features when only training BN (we refer it as random features in below for simplification) have good enough expressive power for image classification tasks. In this section, we mainly focus on verifying that the expressive power of random features is also good enough for differentiable architecture search. Firstly, we provide analysis for exploring the failure of DARTS when we train all the weights of supernet in Sec. 4.1. Secondly, we provide theoretical analyses to explain how random features solve the failure of DARTS in Sec. 4.2. Finally, we conduct experiments to verify the above analyses in Sec. 4.3.

4.1. The devil of skip-connection as an auxiliary connection in the supernet of DARTS

DARTS suffers from the *performance collapse* mainly due to the *instability* optimization when the search space includes skip-connection. For example, Amended-DARTS [2] finds that the normal cell will collapse to choose only skip-connection operation in every edge after a very long time search, like 200 epochs. Similarly, DARTS- [10] also thinks that DARTS tenders to select the skip-connection in the final architecture, since skip-connection can help the back-propagation of the gradient when the supernet is very deep. To verify it, they introduce an extra trainable coefficient $\lambda \in [0, 1]$ on all skip-connections of ResNet-50 [20] and observe that the coefficient will converge to 1 no matter the initialization (see Fig. 3a). Theoretically, given a deep residue model with L residue blocks, the output $X_{l+1} \in \mathbb{R}^{N \times C}$ is calculated as:

$$X_{l+1} = f_l(X_l, W_l) + \lambda X_l, \tag{9}$$

where $X_l \in \mathbb{R}^{N \times C}$ is the input feature and f_l is the non-linear function with weights $W_l \in \mathbb{R}^{C \times C}$. They investigate the gradient of X_l w.r.t. the loss $\mathcal{L}$, which can calculate as ($\mathbb{1}$ is the identity matrix):

$$\frac{\partial \mathcal{L}}{\partial X_l} = \left(\frac{\partial f_l}{\partial X_l} + \lambda \cdot \mathbb{1}\right) \cdot \frac{\partial \mathcal{L}}{\partial X_{l+1}}. \tag{10}$$

With the above gradient formulation, the deep model prefers to push the coefficient λ to 1 for better gradient back-propagation. Thereby, the skip-connection not only serves as a *candidate operation* but also serves as an *auxiliary connection* for stabilizing training in the supernet of DARTS. With the above two roles of skip-connection, it is hard to determine the best sub-network since the supernet will tend to choose skip-connection as the candidate operation.

4.2. The power of random features for diluting skip-connection's role of auxiliary connection

To resolve the search performance collapse of DARTS, it is vital to remove the unfair advantages of skip-connection. DARTS- introduces an auxiliary skip-connection to distinguish the two roles that skip-connection plays in supernet training, thus reserving only its role as candidate operation. In this paper, we focus on using random features to resolve the search performance collapse of DARTS. Through the analysis below, we find that the gradient vanishing problem of the deep model will not occur when we use random features. In this way, there is no need for skip-connection to solve the gradient vanishing problem, but only as of the specific Conv operation with identity kernel. Thereby, random features can help dilute the role of auxiliary connection that skip-connection plays and thus make a fairer competition with all operations. The analysis is shown below.

To understand how random features solve the gradient vanishing problem, we investigate the variance of the gradient in each layer of the plain network. Following the derivation of [19], we assume each layer with one Conv layer and

the activation function g, then the output $X_{l+1} \in \mathbb{R}^{C\times C}$ is:

$$X_{l+1} = g(Y_l), Y_l = W_l X_l, \tag{11}$$

where $X_l \in \mathbb{R}^{k^2C\times 1}$ represents $k\cdot k$ pixels in C input channels, and $W \in \mathbb{R}^{C\times k^2C}$ is the Conv kernal with spatial size k. Then gradient of X_l w.r.t loss $\mathcal{L}$ and its variance can be represented as (ΔX and ΔY donate gradients $\frac{\partial \mathcal{L}}{\partial X}$ and $\frac{\partial \mathcal{L}}{\partial Y}$):

$$\Delta X_l = W_l \Delta Y_l, \tag{12}$$
$$Var[\Delta X_l] = n \cdot Var[W_l] \cdot Var[\Delta Y_l], \tag{13}$$

where $n = k^2C$ and $\Delta Y_l = g'(Y_l)\Delta X_{l+1}$. With the assumption of g is ReLU (thus $g'(Y_l)$ is zero or one with equal probabilities), we have $Var[\Delta Y_l] = \frac{1}{2}Var[\Delta X_{l+1}]$. Putting together the following layers for layer o, we have:

$$Var[\Delta X_o] = Var[\Delta X_{L+1}](\prod_{l=o}^{L} \frac{1}{2} n Var[W_l]). \tag{14}$$

Kaiming initialization [19] gives a sufficient initialization condition ($\frac{1}{2}nVar[W_l] = 1, \forall l$) to make sure the variance of gradient at initialization not exponentially large/small, even when the model is extremely deep. Nevertheless, when we train the weights W of deep models using stochastic gradient descent, variances of weights W will change along with the training procedure, which thereby still cannot avoid the gradients vanishing problem for deep models.

On the contrary, by utilizing random features which freezes all Conv weights of the network (without BN in network) at initialization, the variances of gradient will not change, and the gradient vanishing problem is also avoided. However, the above derivation does not consider the influence of BN. When we consider the utilization of BN, the variances of the gradient are still changing along with the running variance updating. Despite all this, we find that by fixing the weights of Conv and only updating the affine weights of BN, it is still much easy to keep the variances of gradients in a normal range compared with the standard training setting. The derivation is shown as below.

We further include the BN in each layer. Therefore, the output $X_{l+1} = g(Y_l)$ can be represented as:

$$Y_l = \text{BN}(\text{Conv}(X_l, W_l), \{\gamma_l^{\text{bn}}, \beta_l^{\text{bn}}\}) \tag{15}$$
$$= \frac{W_l X_l - \mu_l}{\sqrt{\sigma_l^2 + \epsilon}} \cdot \gamma_l^{\text{bn}} + \beta_l^{\text{bn}}, \tag{16}$$

where $\mu_l \in \mathbb{R}^{C\times 1}$ and $\sigma_l^2 \in \mathbb{R}^{C\times 1}$ are the means and the variances of $W_l X_l$ respectively, and $\gamma_l^{\text{bn}} \in \mathbb{R}^{C\times 1}$ and $\beta_l^{\text{bn}} \in \mathbb{R}^{C\times 1}$ are the learnable affine weights of BN. Here, we let the μ_l and σ_l^2 are the population statistics rather than the mini-batch statistics for simplicity. Thereby, the gradient of X_l w.r.t. loss $\mathcal{L}$ can be denoted as:

$$\Delta X_l = \frac{W_l}{\sqrt{\sigma_l^2 + \epsilon}} \cdot \gamma_l^{\text{bn}} \cdot \Delta Y_l \simeq W_l \cdot \frac{\gamma_l^{\text{bn}}}{\sigma_l} \cdot \Delta Y_l. \tag{17}$$

Then the variance of the gradient ΔX_l will be:

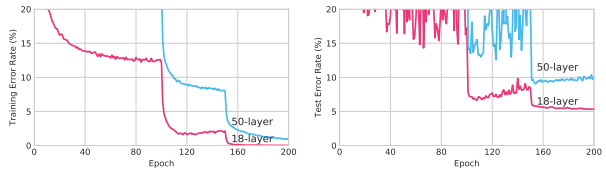
$$Var[\Delta X_l] = n \cdot Var[W_l \cdot \frac{\gamma_l^{\text{bn}}}{\sigma_l}] \cdot Var[\Delta Y_l]. \tag{18}$$

Thereby, to make the gradient not vanish, we consider the similar condition $\frac{1}{2}n \cdot Var[W_l \cdot \frac{\gamma_l^{\text{bn}}}{\sigma_l}] = 1$ like the kaiming initalization. Since we have initialized the weight W_l using xavier [17] or kaiming initialization [19], we only need to make the $W_l \cdot \frac{\gamma_l^{\text{bn}}}{\sigma_l}$ have variance as W_l to keep every layer with a normal range of gradient. Therefore, it is easy and enough to only update the weights of BN to avoid the gradient vanishing problem of deep model.

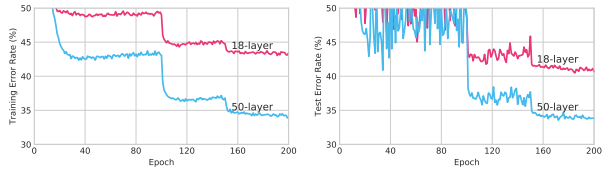
4.3. Verification of random features's power through experiments

As we all know, the deeper models suffer from the performance degradation problem: the network depth increasing leads to the performance degradation. Here, we conduct an intuitive experiment to verify that the deep model with random features will not occur the performance degradation problem since it has solved gradient vanishing problem (Sec. 4.2). We train 18-layer and 50-layer plain ResNets (without skip-connection in the building blocks) on CIFAR-10 with the standard training setting (Conv+BN) and the only BN training setting (Only BN). As shown in Fig. 2a, both the training and test error rate of 50-layer plain ResNet is higher than the 18-layer one when we use the standard training setting. On the contrary, the deeper network has a lower training error rate and test error rate (Fig. 2b) when we use random features. With the above observations, we can conclude that random features can avoid the performance degradation problem of the deep model directly.

Skip-connection introduced by ResNet has been the essential technique for solving the performance degradation problem. With the utilization of random features, we avoid the performance degradation problem. In this way, the supernet will have no preference of choosing skip-connection. Thereby, random features can dilute the role of auxiliary connection that skip-connection plays. We follow the setting of DARTS- for training ResNet-50 with CIFAR-10 to verify it. We visualize changes of λ for skip-connection of ResNet-50 with different initialization . As shown in Fig. 3a, when we train all the trainable weights including the weights of Conv and BN, λ converges to 1 for all initialization settings. On the contrary, we find that λ does not have an evident tendency when we train only the weights of BN. As shown in Fig. 3b, except the begin epochs λ changing frequently, λ keeps constant in the end 100 epochs.

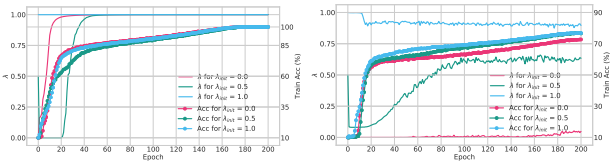


(a) Conv+BN



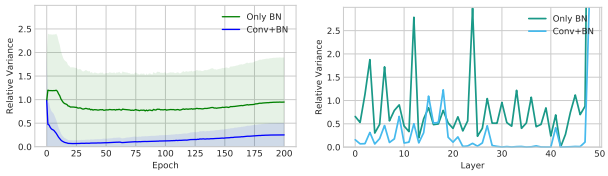
(b) Only BN

Figure 2. Comparison the training and test error rate of two training settings on CIFAR-10 dataset with 18-layer and 50-layer "plain" ResNet (without skip-connection in the building blocks).



(a) Conv+BN (b) Only BN

Figure 3. Visualization of evolved λ with two training settings.



(a) η along model training (b) η when model converges

Figure 4. Visualization of relative variance η on plain ResNet-50.

Finally, we observe the evidence of random features solving the gradient vanishing problem. We calculate $Var(W_l \cdot \frac{\gamma_l^{\text{bn}}}{\sigma_l})$ and $Var(W_l^{init})$ on plain ResNet-50. Then, we plot the relative variance $\eta = Var[W_l \cdot \frac{\gamma_l^{\text{bn}}}{\sigma_l}]/Var[W_l^{init}]$ in Fig. 4. When we train only the weights of BN, the mean of relative scale is close to 1. In this way, the gradients vanishing problem will not happen. However, the relative scale is close to 0 when we train all trainable weights, which makes the gradient vanish for the extremely deep model.

The above phenomenons demonstrate that random features can avoid the gradient vanishing problem, and skip-connection is not necessary for the training of supernet with random features. Thereby, we can diminish the role of skip-connection for avoiding the gradient vanishing of supernet and make supernet focusing on fairer operation selection.

5. Experiment

Based on the above theoretical analyses, we further evaluate DARTS with random features across three popular search spaces. In NAS-Bench-201 [14] (Sec. 5.1), we compare RF-DARTS with the concurrent work BN-NAS, visualize the architecture parameters in RF-DARTS and further make comparisons with the state-of-the-art methods. In DARTS search space [27] (Sec. 5.2), RF-DARTS searches on CIFAR-10 and then searched architectures are transferred to CIFAR-100 and ImageNet. RF-PCDARTS directly searches on ImageNet. In RobustDARTS S1-S4 search spaces [41] (Sec. 5.3), we verify the robustness of RF-DARTS. RF-DARTS and RF-PCDARTS follow the default training setting as DARTS and PC-DARTS respectively. We describe implementation details and plot searched architectures in the supplementary material.

5.1. NAS-Bench-201 results

Comparison with BN-NAS. BN-NAS [6] has introduced a similar supernet optimization paradigm of only training BatchNorm as RF-DARTS. BN indicator for predicting performance is the key contribution in BN-NAS, and only training BN aims to obtain the bonus of 20% search efficiency improvement. Instead, RF-DARTS intends to investigate the expressive power of supernet and alleviate the problem of performance collapse. Except for the different motivations of these works, we further compare the search performance of RF-DARTS and BN-NAS in NAS-Bench-201. As described in BN-NAS [6], BN-indicator for an operation is dependent on the BatchNorm (BN) layer, but there is no BN in none-parametric operations, like skip-connection, avg-pooling, and none. To make BN-NAS generalize to NAS-Bench-201, we introduce additional a BN layer after each none-parametric operation. We train BN-NAS with convolution and BN or only with BN layer with 10 epochs, 50 epochs, and 250 epochs and use evolution search with BN indicator. We conduct each experiment in 3 runs and report the best accuracy in Tab. 3. BN-NAS converges to search cell with six none operations in 3 out of 6 configurations and obtains inferior performance less than 93% in other three configurations, while RF-DARTS achieves the nearest optimal test accuracy **94.36%**. These comparison results imply that RF-DARTS surpasses BN-NAS by a large margin and it is hard to extend BN-NAS to search space with none-parametric operations.

Visualization of architecture parameters. As discussed in Sec. 4, we attribute the success of RF-DARTS to alleviate the issue of skip-connection dominance. To further support our analysis in the practical search phase, we visualize the evolved architecture parameters of best searched architecture on CIFAR-10. As Fig. 5 shows, after searching 50 epochs, the evolved architecture parameters converge

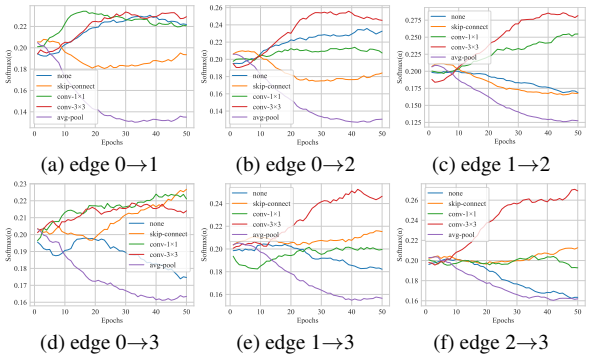
Method	CIFAR-10		CIFAR-100		ImageNet16-120	
	Valid Acc (%)	Test Acc (%)	Valid Acc (%)	Test Acc (%)	Valid Acc (%)	Test Acc (%)
RandomNAS [22]	80.42±3.58	84.07±3.61	52.12±5.55	52.31±5.77	27.22±3.24	26.28±3.09
DARTS^{1st} [27]	39.77±0.00	54.30±0.00	15.03±0.00	15.61±0.00	16.43±0.00	16.32±0.00
DARTS^{2nd} [27]	39.77±0.00	54.30±0.00	15.03±0.00	15.61±0.00	16.43±0.00	16.32±0.00
SETN [12]	84.04±0.28	87.64±0.00	58.86±0.06	59.05±0.24	33.06±0.02	32.52±0.21
GDAS [13]	89.89±0.08	93.61±0.09	71.34±0.04	70.70±0.30	41.59±1.33	41.71±0.98
iDARTS [44]	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- [10]	91.03±0.44	93.80±0.40	71.36±1.51	71.53±1.51	44.87±1.46	45.12±0.82
RF-DARTS (ours)	**91.30±0.36**	**94.27±0.15**	**72.95±0.76**	**72.94±0.81**	**46.40±0.04**	**46.10±0.34**
optimal	91.61	94.37	73.49	73.51	46.77	47.31

Table 2. Search performance on NAS-Bench-201 across CIFAR-10, CIFAR-100 and ImageNet16-120.

Method	Configurations			CIFAR-10 (%)	
	Conv	BN	Epochs	Valid Acc	Test Acc
BN-NAS [6]	✓	✓	10	9.71	10.00
	✓	✓	50	86.28	88.77
	✓	✓	250	88.48	92.28
	✗	✓	10	9.71	10.00
	✗	✓	50	9.71	10.00
	✗	✓	250	84.35	86.45
RF-DARTS	✗	✓	50	**91.55**	**94.36**

Table 3. Comparison with BN-NAS searched on CIFAR-10.

to architecture {edge 0→1: conv-3x3, edge 0→2: conv-3x3, edge 1→2: conv-3x3, edge 0→3: skip-connection, edge 1→3: conv-3x3, edge 2→3: conv-3x3}. There is only one skip-connection in the searched architecture rather than six skip-connection as DARTS^{1st} and DARTS^{2nd}. As for the performance, RF-DARTS boosts test accuracy from 54.30% to 94.36% (**+40.06%**) on CIFAR-10, 15.61% to 73.51% (**+57.90%**) on CIFAR-100 and 16.32% to 46.34% (**+30.02%**) on ImageNet16-120. Thus we further conclude that RF-DARTS dilutes role of skip-connection in supernet optimization and improves the search efficacy by a large margin by making a fairer comparison.



(a) edge 0→1 (b) edge 0→2 (c) edge 1→2

(d) edge 0→3 (e) edge 1→3 (f) edge 2→3

Figure 5. Evolved architecture parameters on CIFAR-10. There are total 6 edges in each search cell. We use edge i→j to represent the operation from source node i to target node j.

Comparison with state-of-the-art methods. For a further step, we verify the efficacy of RF-DARTS with the comparison of state-of-the-art DARTS variants in NAS-Bench-201. RF-DARTS searches on CIFAR-10 with three runs and then we look up the ground truth performance (both validation accuracy and test accuracy) of searched architectures across CIFAR-10, CIFAR-100 and ImageNet16-120. We report results with the mean±std format in Tab. 2. RF-DARTS almost achieves the optimal performance. More specifically, for test accuracy, RF-DARTS outperforms the newest state-of-the-art method DARTS- by **0.47%**, **1.41%** and **0.98%** on CIFAR-10, CIFAR-100, and ImageNet16-120 respectively.

5.2. DARTS search space results

Results searched on CIFAR-10. In DARTS search space, RF-DARTS searches architectures on CIFAR-10 and then evaluates the searched architectures on CIFAR-10, CIFAR-100, and ImageNet. As shown in Tab. 3, RF-DARTS obtains test error 2.60%, 16.50%, and 24.0% on these three datasets respectively. Compared with vanilla DARTS^{2nd}, RF-DARTS achieves 0.16%, 0.96%, and 2.9% improvements across three datasets. As for the comparison with the state-of-the-art method PDARTS, except for the inferior performance in CIFAR-10, RF-DARTS can also obtain 0.13% and 0.4% improvements on the other two datasets. Besides, RF-DARTS obtains comparable test error compared with DARTS- on CIFAR-10, but RF-DARTS has a much stronger transferring ability. To the best of our knowledge, **24.0%** test error on ImageNet is the newest state-of-the-art result with 600M FLOPs constrain when transferring architectures from CIFAR-10.

Results searched on ImageNet. We verify the search efficacy of RF-PCDARTS on ImageNet. To overcome the challenge of huge GPU memory requirements in vanilla DARTS, there are total three popular paradigms that can directly searched on ImageNet in DARTS search space, namely the *partial-channel* paradigm, the *single-path* paradigm and the *training-free* paradigm. We select two representative methods from each paradigm as the baselines: 1) PC-DARTS and DARTS+ belong the

Method	Test error (%)			Params (M)	FLOPs (M)	Search Method
	CIFAR-10	CIFAR-100	ImageNet			
NASNet-A [50]	2.65	17.81	26.0/8.4	3.3	564	RL
PNAS [26]	3.41±0.09	17.63	25.8/8.1	3.2	588	SMBO
AmoebaNet-A [30]	3.34±0.06	-	25.5/8.0	3.2	555	EA
ENAS [28]	2.89	18.91	-	4.6	-	RL
EN^2AS [43]	2.61±0.06	16.45	26.7/8.9	3.1	506	EA
RandomNAS [22]	2.85±0.08	17.63	27.1	4.3	613	random
NSAS [42]	2.59±0.06	17.56	25.5/8.2	3.1	506	random
PARSEC [5]	2.86±0.06	-	26.3	3.6	509	gradient
SNAS [34]	2.85±0.02	20.09	27.3/9.2	2.8	474	gradient
SETN [12]	2.69	17.25	25.7/8.0	4.6	610	gradient
MdeNAS [47]	2.55	17.61	25.5/7.9	3.6	506	gradient
GDAS [13]	2.93	18.38	26.0/8.5	3.4	545	gradient
PDARTS [9]	2.50	16.63	24.4/7.4	3.4	557	gradient
PC-DARTS [36]	2.57±0.07	17.11	25.1/7.8	3.6	586	gradient
DARTS ^{2nd} [27]	2.76±0.09	17.54	26.9/8.7	3.4	574	gradient
DARTS-† [10]	2.58	16.80	25.4/7.9	3.5	547	gradient
RF-DARTS (ours)	2.60	**16.50**	**24.0/7.2**	4.6	593	gradient

Table 4. Comparison results with state-of-the-art weight-sharing NAS approaches in DARTS search space. These evaluated architectures are searched on CIFAR-10 and transferred to CIFAR-100 and ImageNet. † we retrain the architecture reported in DARTS- [10].

Paradigm	Method	Test error (%)	Params (M)	FLOPs (M)
single-path	SPOS [18]	25.5/7.9	4.6	512
	RLNAS [46]	24.1/7.1	5.5	597
training-free	FreeNAS [45]	25.1/7.8	3.6	592
	TE-NAS [7]	24.5/7.5	5.4	591
partial-channel	PC-DARTS [36]	24.2/7.3	5.3	597
	DARTS+ [23]	23.9/7.4	5.1	582
	RF-PCDARTS	**23.9/7.1**	5.4	600

Table 5. Comparison results with popular NAS paradigms directly searched on ImageNet (mobile setting) in DARTS search space.

Benchmark		DARTS	R-DARTS		DARTS		Ours
			DP	L2	ES	ADA	
C10	S1	3.84	3.11	**2.78**	3.01	3.10	2.95
	S2	4.85	3.48	3.31	**3.26**	3.35	4.21
	S3	3.34	2.93	**2.51**	2.74	2.59	2.83
	S4	7.20	3.58	3.56	3.71	4.84	**3.33**
C100	S1	29.46	25.93	24.25	28.37	24.03	**22.75**
	S2	26.05	22.30	22.24	23.25	23.52	**22.18**
	S3	28.90	**22.36**	23.99	23.73	23.37	24.67
	S4	22.85	22.18	21.94	21.26	23.20	**21.19**
SVHN	S1	4.58	2.55	4.79	2.72	2.53	**2.42**
	S2	3.53	2.52	2.51	2.60	2.54	**2.39**
	S3	3.41	2.49	2.48	2.50	2.50	**2.47**
	S4	3.05	2.61	2.50	2.51	**2.46**	2.49

Table 6. Test error on CIFAR-10, CIFAR-100 and SVHN across RobustDARTS S1-S4. Both RF-DARTS and three benchmarks search directly on target datasets.

partial-channel paradigm; 2) SPOS and RLNAS belong the *single-path* paradigm; 3) FreeNAS and TE-NAS follow the *training-free* paradigm. Following the setting of PC-DARTS, RF-PCDARTS directly searches architectures on ImageNet. As Tab. 4 shows, considering top-1 and top-5 test error comprehensively, RF-PCDARTS consistently exceeds six powerful benchmarks across three paradigms.

5.3. Robustness verification across DARTS S1-S4

R-DARTS [41] proposes four search spaces S1-S4 where vanilla DARTS fails. We evaluate the robustness of RF-DARTS in S1-S4 across CIFAR-10, CIFAR-100, and SVHN. In this section, RF-DARTS directly searches architectures on target datasets. To prove the effectiveness of RF-DARTS, we choose DARTS, R-DARTS(DP/L2), and DARTS(ES/ADA) as benchmarks. Tab. 5 shows comparison results between RF-DARTS and three benchmarks. RF-DARTS consistently outperforms vanilla DARTS by a large margin across four search spaces and three datasets. For the other two stronger elaborate benchmarks R-DARTS(DP/L2) and DARTS(ES/ADA), RF-DARTS still obtain superior performance in most cases. These results demonstrate that RF-DARTS performs substantially robust.

6. Conclusion

To alleviate performance collapse in DARTS, we challenge the conventional supernet optimization paradigm and demystify DARTS from a new perspective of expressive power in supernet. We surprisingly find that only training BatchNorm achieves higher search performance, which surpasses DARTS with other three learnable module combinations by a large margin and thus we propose RF-DARTS. We further theoretically analyze the variance of gradient in RF-DARTS and conclude that random features can alleviate the problem of performance collapse by diminishing the role of skip-connection. Comprehensive experiments across various datasets and search spaces consistently demonstrate the effectiveness and robustness of RF-DARTS.

References

[1] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. *arXiv preprint arXiv:1611.02167*, 2016. 1

[2] Kaifeng Bi, Changping Hu, Lingxi Xie, Xin Chen, Longhui Wei, and Qi Tian. Stabilizing DARTS with amended gradient estimation on architectural parameters. *CoRR*, 2019. 3, 4

[3] Hans-Dieter Block. The perceptron: A model for brain functioning. *Reviews of Modern Physics*, 34(1):123, 1962. 3

[4] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. In *ICLR*, 2019. 3

[5] Francesco Paolo Casale, Jonathan Gordon, and Nicolo Fusi. Probabilistic neural architecture search. *arXiv preprint arXiv:1902.05116*, 2019. 8

[6] Boyu Chen, Peixia Li, Baopu Li, Chen Lin, Chuming Li, Ming Sun, Junjie Yan, and Wanli Ouyang. Bn-nas: Neural architecture search with batch normalization. In *ICCV*, 2021. 3, 6, 7

[7] Wuyang Chen, Xinyu Gong, and Zhangyang Wang. Neural architecture search on imagenet in four GPU hours: A theoretically inspired perspective. In *ICLR*, 2021. 1, 8

[8] Xiangning Chen and Cho-Jui Hsieh. Stabilizing differentiable architecture search via perturbation-based regularization. In *ICML*, 2020. 3

[9] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *ICCV*, 2019. 1, 2, 3, 8

[10] Xiangxiang Chu, Xiaoxing Wang, Bo Zhang, Shun Lu, Xiaolin Wei, and Junchi Yan. DARTS-: robustly stepping out of performance collapse without indicators. In *ICLR*, 2021. 1, 2, 3, 4, 7, 8

[11] Xiangxiang Chu, Tianbao Zhou, Bo Zhang, and Jixiang Li. Fair DARTS: eliminating unfair advantages in differentiable architecture search. In *ECCV*, 2020. 3

[12] Xuanyi Dong and Yi Yang. One-shot neural architecture search via self-evaluated template network. In *ICCV*, 2019. 7, 8

[13] Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four GPU hours. In *CVPR*, 2019. 1, 3, 7, 8

[14] Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *ICLR*, 2020. 2, 6

[15] Jonathan Frankle, David J. Schwab, and Ari S. Morcos. Training batchnorm and only batchnorm: On the expressive power of random features in cnns. In *ICLR*, 2021. 3, 4

[16] Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Linearized two-layers neural networks in high dimension. *CoRR*, 2019. 3

[17] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010. 5

[18] 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 *ECCV*, 2020. 3, 8

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015. 4, 5

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 4

[21] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 2, 3

[22] Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. In *Uncertainty in artificial intelligence*, 2020. 7, 8

[23] 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, 3, 8

[24] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *CVPR*, 2019. 3

[25] Chenxi Liu, Piotr Dollár, Kaiming He, Ross Girshick, Alan Yuille, and Saining Xie. Are labels necessary for neural architecture search? In *ECCV*, 2020. 1

[26] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *ECCV*, 2018. 8

[27] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: differentiable architecture search. In *ICLR*, 2019. 1, 2, 3, 6, 7, 8

[28] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In *ICML*, 2018. 3, 8

[29] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *NeurIPS*, 2007. 3

[30] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *AAAI*, 2019. 8

[31] Alvin Wan, Xiaoliang Dai, Peizhao Zhang, Zijian He, Yuandong Tian, Saining Xie, Bichen Wu, Matthew Yu, Tao Xu, Kan Chen, Peter Vajda, and Joseph E. Gonzalez. Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions. In *CVPR*, 2020. 3

[32] Ruochen Wang, Minhao Cheng, Xiangning Chen, Xiaocheng Tang, and Cho-Jui Hsieh. Rethinking architecture selection in differentiable NAS. In *ICLR*, 2021. 1

[33] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *CVPR*, 2019. 3

[34] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. SNAS: stochastic neural architecture search. In *ICLR*, 2019. 3, 8

[35] Hang Xu, Lewei Yao, Zhenguo Li, Xiaodan Liang, and Wei Zhang. Auto-fpn: Automatic network architecture adaptation for object detection beyond classification. In *ICCV*, 2019. 3

[36] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. PC-DARTS: partial channel connections for memory-efficient architecture search. In *ICLR*, 2020. 1, 2, 3, 8

[37] Yibo Yang, Hongyang Li, Shan You, Fei Wang, Chen Qian, and Zhouchen Lin. ISTA-NAS: efficient and consistent neural architecture search by sparse coding. In *NeurIPS*, 2020. 3

[38] Yibo Yang, Shan You, Hongyang Li, Fei Wang, Chen Qian, and Zhouchen Lin. Towards improving the consistency, efficiency, and flexibility of differentiable neural architecture search. In *CVPR*, 2021. 3

[39] Gilad Yehudai and Ohad Shamir. On the power and limitations of random features for understanding neural networks. In *NeurIPS*, 2019. 3

[40] Hongyuan Yu and Houwen Peng. Cyclic differentiable architecture search. *arXiv preprint arXiv:2006.10724*, 2020. 3

[41] Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. Understanding and robustifying differentiable architecture search. In *ICLR*, 2020. 1, 2, 3, 6, 8

[42] Miao Zhang, Huiqi Li, Shirui Pan, Xiaojun Chang, Zongyuan Ge, and Steven W Su. Differentiable neural architecture search in equivalent space with exploration enhancement. In *NeurIPS*, 2020. 8

[43] Miao Zhang, Huiqi Li, Shirui Pan, Taoping Liu, and Steven W Su. One-shot neural architecture search via novelty driven sampling. In *IJCAI*, 2020. 8

[44] 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. 7

[45] Miao Zhang, Steven Su, Shirui Pan, Xiaojun Chang, Wei Huang, and Gholamreza Haffari. Differentiable architecture search without training nor labels: A pruning perspective. *arXiv preprint arXiv:2106.11542*, 2021. 1, 8

[46] Xuanyang Zhang, Pengfei Hou, Xiangyu Zhang, and Jian Sun. Neural architecture search with random labels. In *CVPR*, 2021. 1, 8

[47] Xiawu Zheng, Rongrong Ji, Lang Tang, Baochang Zhang, Jianzhuang Liu, and Qi Tian. Multinomial distribution learning for effective neural architecture search. In *ICCV*, 2019. 8

[48] Qinqin Zhou, Xiawu Zheng, Liujuan Cao, Bineng Zhong, Teng Xi, Gang Zhang, Errui Ding, Mingliang Xu, and Rongrong Ji. Ec-darts: Inducing equalized and consistent optimization into darts. In *ICCV*, 2021. 3

[49] Yichen Zhu and Xiaowei Fu. Bnnas++: Towards unbiased neural architecture search with batch normalization. *IEEE Access*, 10:128424–128432, 2022. 3

[50] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016. 1, 8

[51] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *CVPR*, 2018. 1