Noisy Differentiable Architecture Search

Xiangxiang Chu
chuxiangxiang@xiaomi.com

Bo Zhang
zhangbo11@xiaomi.com

Xiaomi AI Lab
Beijing, China

Abstract

Simplicity is the ultimate sophistication. Differentiable Architecture Search (DARTS) has now become one of the mainstream paradigms of neural architecture search. However, it largely suffers from the well-known performance collapse issue due to the aggregation of skip connections. It is thought to have overly benefited from the residual structure which accelerates the information flow. To weaken this impact, we propose to inject unbiased random noise to impede the flow. We name this novel approach NoisyDARTS. In effect, a network optimizer should perceive this difficulty at each training step and refrain from overshooting, especially on skip connections. In the long run, since we add no bias to the gradient in terms of expectation, it is still likely to converge to the right solution area. We also prove that the injected noise plays a role in smoothing the loss landscape, which makes the optimization easier. Our method features extreme simplicity and acts as a new strong baseline. We perform extensive experiments across various search spaces, datasets, and tasks, where we robustly achieve state-of-the-art results. Our code is available here.

1 Introduction

Differentiable architecture search [32] suffers from a well-known performance collapse issue noted by [5, 7]. Namely, while the over-parameterized model is well optimized, its inferred model tends to have an excessive number of skip connections, which dramatically degrades the searching performance. Quite an amount of previous research has focused on addressing this issue [5, 7, 25, 28, 51]. Among them, Fair DARTS [7] concludes that it is due to an unfair advantage in an exclusively competitive environment. Under this perspective, early-stopping methods like [28, 51] or greedy pruning [25] can be regarded as means to prevent such unfairness from overpowering. However, the one-shot network is generally not well converged if halted too early, which gives low confidence to derive the final model.

More precisely, most of the existing approaches [5, 25, 51] addressing the fatal collapse can be categorized within the following framework: first, characterize the outcome when the collapse occurs (e.g., larger Hessian eigenvalue as in RobustDARTS [51] or too many skip connections in a cell [5, 28]), and then carefully design various criteria to avoid stepping into it. There are two main drawbacks of these methods. One is that the search results heavily rely on the validity of human-designed criteria, otherwise, inaccurate criteria may reject good
models (see Section 5). The other is that these criteria only force the searching process to stay away from a bad solution. However, the goal of neural architecture search is not just to avoid bad solutions but to robustly find much better ones.

Our contributions can be summarized in the following,

- Other than designing various criteria, we demonstrate a simple but effective approach to address the performance collapse issue in DARTS. Specifically, we inject various types of independent noise into the candidate operations to make good ones robustly win. This approach also has an effect of smoothing loss landscape.
- We prove that the required characteristics of the injected noise should be unbiased and of moderate variance. Furthermore, it is the unbiasedness that matters, not a specific noise type. Surprisingly, our well-performing models are found with rather high Hessian eigenvalues, disproving the need for the single-point Hessian norm as an indicator of the collapse [51], since it can’t describe the overall curvatures of its wider neighborhood.
- Extensive experiments performed across various search spaces (including the more difficult ones proposed in [51] and datasets (15 benchmarks in total) show that our method can address the collapse effectively. Moreover, we robustly achieve state-of-the-art results with 3× fewer search costs than RobustDARTS.

2 Related work

Differentiable architecture search DARTS [32] has widely disseminated the paradigm of solving architecture search with gradient descent [1, 34, 47]. It constructs an over-parameterized supernet incorporating all the choice operations. Each discrete choice is assigned with a continuous architectural weight \( \alpha \) to denote its relative importance, and the outputs of all the paralleling choices are summed up using a softmax function \( \sigma \). Through iterative optimization of supernet parameters and architectural ones, competitive operations are supposed to stand out with the highest \( \sigma(\alpha) \) to be chosen to derive the final model. Though being efficient, it is known unstable to reproduce [50].

Endeavors to improve the performance collapse in DARTS Several previous works have focused on addressing the collapse. For instance, P-DARTS [5] point out that DARTS gradually leans towards skip connection operations since they ease the training. However, while being parameter-free, they are essentially weak to learn visual features which lead to degenerate performance. To resolve this, they proposed to drop out paths through skip connections with a decay rate. Still, the number of skip connections in normal cells varies, for which they impose a hard-coded constraint, limiting this number to be \( M \). Later DARTS+ [28] simply early stops when there are exactly two skip connections in a cell. RobustDARTS [51] discovers degenerate models (where skip connections are usually dominant) correlate with increasingly large Hessian eigenvalues, for which they utilize an early stopping strategy while monitoring these values.

3 Noisy DARTS

3.1 Motivation

We are motivated by two distinct and orthogonal aspects of DARTS: how to make the optimization easier and how to remove the unfair competition from candidate operations.
Smooth loss landscape helps stochastic gradient optimization (SGD) to find the solution path at early optimization stages. SGD can escape local minima to some extent \cite{22} but still have difficulty navigating chaotic loss landscapes \cite{26}. Combining it with a smoother loss function can relieve the pain for optimization which leads to better solutions \cite{15, 26}. Previously, RobustDARTS \cite{51} empirically finds that the collapse is highly related to the sharp curvature of the loss w.r.t $\alpha$, for which they use Hessian eigenvalues as an indicator of the collapse. However, this indirect indicator at a local minimum fails to characterize its relatively larger neighborhood, which we discuss in detail in Section 5. Therefore, we are driven to contrive a more direct and effective way to smooth the landscape.

Adding noise is a promising way to smooth the loss landscape. Random noises are used to boost adversarial generalization by \cite{17, 24, 33}. A recent study by \cite{45} points out that a flatter adversarial loss landscape is closely related to better generalization. This leads us to first reformulate DARTS from the probabilistic distribution’s perspective as follows,

$$\alpha^* = \arg\min_{\alpha} L_{\alpha}(w, z) = \arg\max_{\alpha} \mathbb{E}_{x, y \sim P_{\text{val}}; z \sim P(z)} \log P(y|x, w^*, \alpha, z)$$

$$\text{s.t. } w^* = \arg\max_{w} \mathbb{E}_{x, y \sim P_{\text{train}}; z \sim P(z)} \log P(y|x, w, \alpha, z) \quad (1)$$

where $z \sim \delta(z)$. The random variable $z$ is subject to the Dirac distribution and added to the intermediate features. For a multiplicative version, we can simply set $z \sim \delta(z - 1)$. We follow \cite{32} for the rest notations. To incorporate noise and smooth DARTS (Equation 1), we propose a direct approach by setting,

$$z \sim N(\mu, \sigma). \quad (2)$$

We choose additive Gaussian noise for simplicity. Experiments on uniform noise are also provided in Section 3.1 (supplementary). The remaining problem is where to inject the noise and how to calibrate $\mu$ and $\sigma$.

Unfairness of skip connections from fast convergence. Apart from the above-mentioned perspective, we notice from prior work that skip connections are the primary subject to consider \cite{5, 7, 28}. While being summed with other operations, a skip connection builds up a residual structure as in \cite{16}. A similar form is also proposed in highway networks \cite{40}. Such a residual structure is generally helpful for training deep networks, as well as the supernet of DARTS. However, as skip connections excessively benefit from this advantage \cite{5, 7}, it leads us to overestimate its relative importance, while others are under-evaluated. Therefore, it is appropriate to disturb the gradient flow (by injecting noise as a natural choice) right after the intermediate outputs of various candidate operations. In this way, we can regularize the gradient flow from different candidate operations and let them compete in a fair environment. We term this approach NFA, short for “Noise For All”. Considering the unfair advantage is mainly from the skip connection, we can also choose to inject noises only after this operation. We call this approach OFS, short for “Only For Skip”. This option is even simpler than NFA. We use OFS as the default implementation.

3.2 Requirements for the injected noise

A basic and reasonable requirement is that, applying Equation 2 should make a close approximation to Equation 1. Since each iteration is based on backward propagation, we relax this requirement to having an unbiased gradient in terms of its expectation at each iteration. Here we induce the requirement based on OFS for simplicity.
Design of $\mu$ We let $\tilde{x}$ be the noise injected into the skip operation, and $\alpha_s$ be the corresponding architectural weight. The loss of a skip connection operation can be written as,

$$\mathcal{L} = g(y), \quad y = f(\alpha_s) \cdot (x + \tilde{x})$$

(3)

where $g(y)$ is the validation loss function and $f(\alpha_s)$ gives the softmax output for $\alpha_s$. Approximately, when the noise is much smaller than the output features, we have

$$y^* \approx f(\alpha_s) \cdot x \quad \text{when} \quad \tilde{x} \ll x.$$  

(4)

In the noisy scenario, the gradient of the parameters via the skip connection operation becomes,

$$\frac{\partial \mathcal{L}}{\partial \alpha_s} = \frac{\partial \mathcal{L}}{\partial y} \frac{\partial y}{\partial \alpha_s} = \frac{\partial \mathcal{L}}{\partial y} \frac{\partial f(\alpha_s)}{\partial \alpha_s} (x + \tilde{x}).$$

(5)

As random noise $\tilde{x}$ brings uncertainty to the gradient update, skip connections have to overcome this difficulty in order to win over other operations. Their unfair advantage is then much weakened. However, not all types of noise are equally effective in this regard. Formally, the expectation of its gradient can be written as,

$$E[\tilde{x}] [\nabla \alpha_s] = \mathbb{E}[\tilde{x}] \left[ \frac{\partial \mathcal{L}}{\partial y} \frac{\partial f(\alpha_s)}{\partial \alpha_s} (x + \tilde{x}) \right] \approx \frac{\partial \mathcal{L}}{\partial y^*} \frac{\partial f(\alpha_s)}{\partial \alpha_s} (x + \mathbb{E}[\tilde{x}]).$$

(6)

Supposing that $\frac{\partial \mathcal{L}}{\partial y^*}$ is smooth, we can use $\frac{\partial \mathcal{L}}{\partial y^*}$ to approximate the its small neighborhood. Based on the premise stated in Equation 4, we take $\frac{\partial \mathcal{L}}{\partial y^*}$ out of the expectation in Equation 6 to make an approximation. As there is still an extra $\mathbb{E}[\tilde{x}]$ in the gradient of skip connection, to keep the gradient unbiased, $\mathbb{E}[\tilde{x}]$ should be 0. It’s intuitive to see the unbiased injected noise can play a role of encouraging the exploration of other operations.

**Design of $\sigma$** The variance $\sigma^2$ controls the magnitude of the noise, which also represents the strength to step out of local minima. Intuitively, the noise should neither be too big (overtaking) nor too small (ineffective). For simplicity, we start with Gaussian noise and other options are supposed to work as well. Notably, applying Equation 2 when $\sigma=0$ falls back to Equation 1.

### 3.3 Stepping out of the performance collapse by noise

Based on the above analysis, we propose NoisyDARTS to step out of the performance collapse. In practice, we inject Gaussian noise $\tilde{x} \sim N(\mu, \sigma)$ into skip connections to weaken the unfair advantage. Formally, the edge $e_{i,j}$ from node $i$ to $j$ in each cell operates on $i$-th input feature $x_i$ and its output is denoted as $o_{i,j}(x_i)$. The intermediate node $j$ gathers all inputs from the incoming edges: $x_j = \sum_{i<j} o_{i,j}(x_i)$. Let $O = \{o^0_{i,j}, o^1_{i,j}, \ldots, o^{M-1}_{i,j}\}$ be the set of $M$ candidate operations on edge $e_{i,j}$ and specially let $o^0_{i,j}$ be the skip connection $o^{skip}_{i,j}$. NoisyDARTS injects the additive noise $\tilde{x}$ into skip operation $o^{skip}_{i,j}$ to get a mixed output,

$$\bar{o}_{i,j}(x) = \sum_{k=1}^{M-1} f(\alpha_{ok}) o^k(x) + f(\alpha_{oskip}) o^{skip}(x + \tilde{x}).$$

(7)
The architecture search problem remains the same as the original DARTS, which is to alternately learn $\alpha^*$ and network weights $w^*$ that minimize the validation loss $L_{\text{val}}(\alpha^*, w^*)$. To summarize, NoisyDARTS (OFS) is shown in Algorithm 1 (supplementary). The NFA version is in Algorithm 2 (supplementary).

The role of noise. The role of the injected noise is threefold. Firstly, it breaks the unfair advantage so that the final chosen skip connections indeed have substantial contribution for the standalone model. Secondly, it encourages more exploration to escape bad local minima, whose role is akin to the noise in SGLD [52]. Lastly, it smooths the loss landscape w.r.t $\alpha$. If we denote validation loss as $L$, this role can be explained due to the fact that our approach implicitly controls the loss landscape. Supposing that the injected noise $z$ is small and $z \sim N(0, \sigma^2 I)$, the expectation of the loss over $z$ can be approximated by

$$
\mathbb{E}_z [L_v(w, \alpha_s, z)] \approx \mathbb{E}_z[L_v(w, \alpha_s, 0) + \nabla_z L_v(w, \alpha_s, 0)z + \frac{1}{2}z^T \nabla^2_z L_v(w, \alpha_s, 0)z]
$$

$$
= L_v(w, \alpha_s, 0) + \frac{\sigma^2}{2} \text{Tr}\{\nabla^2_z L_v(w, \alpha_s, 0)\}
$$

$$
\approx L_v(w, \alpha_s, 0) + \frac{\beta \sigma^2 \alpha_s^2}{2} \text{Tr}\{\nabla^2_{\alpha_s} L_v(w, \alpha_s, 0)\}
$$

where $z \sim N(0, \sigma^2 I), \beta = \mathbb{E}_{o_{\text{skip}}(x)^T o_{\text{skip}}(x)}^{-1}$, $I$ is unit matrix.

Its role can be better understood via the visualization in Figure 2, where DARTS obtains a sharp landscape with oval contours and ours has round ones.

4 Experiments

4.1 Search spaces and 15 benchmarks

To verify the validity of our method, we adopt several search spaces: the DARTS search space from [4], MobileNetV2’s search space as in [5], four harder spaces (from $S_1$ to $S_4$) from [11]. We use NAS-Bench-201 [14] to benchmark our methods.

DARTS’s search space (Benchmark 1) It consists of a stack of duplicate normal cells and reduction cells, which are represented by a DAG of 4 intermediate nodes. Between every two nodes there are several candidate operations (max pooling, average pooling, skip connection, separable convolution $3 \times 3$ and $5 \times 5$, dilation convolution $3 \times 3$ and $5 \times 5$).

MobileNetV2’s search space (Benchmark 2) It is the same as that in ProxylessNAS [11]. We search proxylessly on ImageNet in this space. It uses the standard MobileNetV2’s backbone architecture [5], which comprises 19 layers and each contains 7 choices: inverted bottleneck blocks denoted as Ex_Ky (expansion rate $x \in \{3, 6\}$, kernel size $y \in \{3, 5, 7\}$) and a skip connection. The stem, the first bottleneck block and the tail is kept unchanged, see Figure 3 (supplementary) for reference.

$S_1$-S$4$ (Benchmark 3-14) These are reduced search spaces introduced by RobustDARTS [12]. $S_1$ is a preoptimized search space with two operation per edge, see [12] for the detail. For each edge in the DAG, $S_2$ has only $\{3 \times 3$ SepConv, SkipConnect$\}$, $S_3$ has $\{3 \times 3$ SepConv, SkipConnect, Zero (None)$\}$, and $S_4$ has $\{3 \times 3$ SepConv, Noise$\}$. We search on three datasets for each search space, which makes 12 benchmarks.
Table 1: Results on CIFAR-10 (left) and ImageNet (right). NoisyDARTS-a and b are the models searched on CIFAR-10 when $\sigma = 0.2$ and $\sigma = 0.1$ respectively (Figure 8 and 9 in the supplementary). NoisyDARTS-A (Figure 3 in the supplementary) is searched on ImageNet in the MobileNetV2-like search space as in [44].

NAS-Bench-201 (Benchmark 15) NAS-Bench-201 [14] is a cell based search space with known evaluations of each candidate architecture, where DARTS severely suffers from the performance collapse. It includes 15625 sub architectures in total. Specifically, it has 4 intermediate nodes and 5 candidate operations (none, skip connection, $1 \times 1$ convolution, $3 \times 3$ convolution and $3 \times 3$ average pooling).

4.2 Datasets
We use a set of standard image classification datasets CIFAR-10, CIFAR-100 [23], SVHN [35] and ImageNet [11] for both searching and training. We also search for GCNs on ModelNet [46] as in [25] (see Section 3.1 in the supplementary).

4.3 Searching Results
Searching on CIFAR-10. In the search phase, we use similar hyperparameters and tricks as [32]. All experiments are done on a Tesla V100 with PyTorch 1.0 [36]. The search phase takes about 0.4 GPU days. We only use the first-order approach for optimization since it is more efficient. The best models are selected under the noise with a zero mean and $\sigma = 0.2$. An example of the evolution of the architectural weights during the search phase is exhibited in Figure 2 (see Section 3 in the supplementary). For training a single model, we use the same strategy and data processing tricks as [5, 32], and it takes about 16 GPU hours. The results are shown in Table 1. The best NoisyDARTS model (NoisyDARTS-a) achieves a new state-of-the-art result of 97.63% with only 534M FLOPS and 3.25M parameters, whose genotypes are shown in Figure 8 (see Section 4 in the supplementary).
Searching in the reduced search spaces of RobustDARTS. We also study the performance of our approach under reduced search spaces, compared with DARTS [32], RDARTS [51] and SDARTS [3]. Particularly we use OFS for S1, S2, and S3 (from [51]), where DARTS severely suffers from the collapse owing to an excessive number of skip connections. For S4, where skip operations are not present, we apply NFA. We kept the same hyper-parameters as [32] for training every single model to make a fair comparison. Since the unfair advantage is intensified in the reduced search spaces, we use stronger Gaussian noise (e.g. $\sigma = 0.6, 0.8$). As before, we don’t utilize any regularization tricks. The results are given in Table 2 and Table 9 (see Section 3.11 in the supplementary). Each search is repeated only three times to obtain the average.

<table>
<thead>
<tr>
<th>Data</th>
<th>Space</th>
<th>DARTS</th>
<th>DARTS$^{\text{ADV}}$</th>
<th>DARTS$^{\text{LS}}$</th>
<th>Ours</th>
<th>RDARTS$^{\text{ADV}}$</th>
<th>SDARTS$^{\text{LS}}$</th>
<th>SDARTS$^{\text{ADV}}$</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>C10</td>
<td>S1</td>
<td>95.34±0.71</td>
<td>96.97±0.08</td>
<td>96.95±0.07</td>
<td>97.05±0.18</td>
<td>97.22</td>
<td>97.22</td>
<td>97.27</td>
<td>97.27</td>
</tr>
<tr>
<td></td>
<td>S2</td>
<td>95.58±0.40</td>
<td>96.41±0.31</td>
<td>96.59±0.14</td>
<td>96.59±0.11</td>
<td>96.69</td>
<td>96.67</td>
<td>96.59</td>
<td>96.71</td>
</tr>
<tr>
<td></td>
<td>S3</td>
<td>95.88±0.85</td>
<td>97.01±0.34</td>
<td>96.29±1.14</td>
<td>97.42±0.08</td>
<td>97.49</td>
<td>97.47</td>
<td>97.51</td>
<td>97.53</td>
</tr>
<tr>
<td></td>
<td>S4</td>
<td>93.05±0.18</td>
<td>96.11±0.67</td>
<td>95.83±0.21</td>
<td>97.22±0.08</td>
<td>96.44</td>
<td>97.07</td>
<td>97.13</td>
<td>97.29</td>
</tr>
<tr>
<td>C100</td>
<td>S1</td>
<td>70.07±0.41</td>
<td>75.06±0.81</td>
<td>71.10±0.51</td>
<td>77.89±0.88</td>
<td>75.75</td>
<td>76.49</td>
<td>77.67</td>
<td>78.83</td>
</tr>
<tr>
<td></td>
<td>S2</td>
<td>71.25±0.12</td>
<td>73.12±1.11</td>
<td>75.32±1.43</td>
<td>78.15±0.44</td>
<td>77.76</td>
<td>77.22</td>
<td>79.44</td>
<td>78.82</td>
</tr>
<tr>
<td></td>
<td>S3</td>
<td>70.99±0.24</td>
<td>75.45±0.63</td>
<td>73.01±1.79</td>
<td>79.48±0.59</td>
<td>76.01</td>
<td>78.91</td>
<td>78.92</td>
<td>79.93</td>
</tr>
<tr>
<td></td>
<td>S4</td>
<td>75.23±1.51</td>
<td>76.34±0.90</td>
<td>76.10±2.01</td>
<td>78.37±0.42</td>
<td>78.06</td>
<td>78.54</td>
<td>78.75</td>
<td>78.84</td>
</tr>
<tr>
<td>SVHN</td>
<td>S1</td>
<td>90.12±5.50</td>
<td>97.41±0.07</td>
<td>97.20±0.09</td>
<td>97.44±0.06</td>
<td>95.21</td>
<td>97.14</td>
<td>97.51</td>
<td>97.51</td>
</tr>
<tr>
<td></td>
<td>S2</td>
<td>96.31±0.12</td>
<td>97.21±0.22</td>
<td>97.32±0.18</td>
<td>97.60±0.08</td>
<td>97.49</td>
<td>97.61</td>
<td>97.65</td>
<td>97.66</td>
</tr>
<tr>
<td></td>
<td>S3</td>
<td>96.00±1.01</td>
<td>97.42±0.07</td>
<td>97.22±0.19</td>
<td>97.58±0.06</td>
<td>97.52</td>
<td>97.64</td>
<td>97.60</td>
<td>97.63</td>
</tr>
<tr>
<td></td>
<td>S4</td>
<td>97.10±0.02</td>
<td>97.48±0.06</td>
<td>97.45±0.15</td>
<td>97.59±0.09</td>
<td>97.50</td>
<td>97.54</td>
<td>97.58</td>
<td>97.67</td>
</tr>
</tbody>
</table>

Table 2: Comparison in the reduced spaces of RobustDARTS [51]. For NoisyDARTS, we use NFA for S4 since there is no skip connection in it, and OFS for all the rest. †: 16 initial channels (retrained). Four right columns are the best out of three runs. ADA: adaptive regularization, ES: early-stop, L2: L2 regularization (ADA, ES, L2 are from RDARTS [51]).

Searching proxylessly on ImageNet. In the search phase, we use $\mu = 0$ and $\sigma = 0.2$ and we don’t optimize the hyper-parameters regarding cost. It takes about 12 GPU days on Tesla V100 machines (more details are included in Section 3.5 in the supplementary). As for training searched models, we use similar training tricks as EfficientNet [41]. The evolution of dominating operations during the search is illustrated in Figure 4 (supplementary). Compared with DARTS (66.4%), the injected noise in NoisyDARTS successfully eliminates the unfair advantage. Our model NoisyDARTS-A (see Figure 3 in the supplementary) obtains the new state of the art results: 76.1% top-1 accuracy on ImageNet with 4.9M number of parameters. After being equipped with more tricks as in EfficientNet, such as squeeze-and-excitation [23] and AutoAugment [5], it obtains 77.9% top-1 accuracy.

Searching on NAS-Bench-201. We report the results (averaged on 3 runs of searching) on NAS-bench-201 [26] in Table 3. Our method surpasses SETN [26] with a clear margin using 3 fewer search cost. This again proves NoisyDARTS to be a robust and powerful method. Learnable and decayed $\sigma$ are used for ablation purposes (see Section 4.4).

Searching GCN on ModelNet10. We follow the same setting as SGAS [22] to search GCN networks on ModelNet10 [26] and evaluate them on ModelNet40. Our models (see Figure 20 in the supplementary) are on par with SGAS as reported in Table 7.
### Table 3: Comparison on NAS-Bench-201. Averaged on 3 searches. The best for is in bold and underlined, while the second best is in bold.

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Dataset</th>
<th>Benchmark</th>
<th>Acc (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NoisyDARTS</td>
<td>w/ Noise</td>
<td>CIFAR-10</td>
<td>1</td>
<td>97.35±0.23</td>
</tr>
<tr>
<td>DARTS</td>
<td>w/o Noise</td>
<td>CIFAR-10</td>
<td>1</td>
<td>96.62±0.23</td>
</tr>
<tr>
<td>NoisyDARTS</td>
<td>w/ Noise</td>
<td>ImageNet</td>
<td>2</td>
<td>76.1</td>
</tr>
<tr>
<td>DARTS</td>
<td>w/o Noise</td>
<td>ImageNet</td>
<td>2</td>
<td>66.4</td>
</tr>
</tbody>
</table>

### Table 4: NoisyDARTS is robust across CIFAR-10 and ImageNet. *: Reported by [50]

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Dataset</th>
<th>Benchmark</th>
<th>Acc (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NoisyDARTS</td>
<td>w/ Noise</td>
<td>CIFAR-10</td>
<td>1</td>
<td>97.35±0.23</td>
</tr>
<tr>
<td>DARTS</td>
<td>w/o Noise</td>
<td>CIFAR-10</td>
<td>1</td>
<td>96.62±0.23</td>
</tr>
<tr>
<td>NoisyDARTS</td>
<td>w/ Noise</td>
<td>ImageNet</td>
<td>2</td>
<td>76.1</td>
</tr>
<tr>
<td>DARTS</td>
<td>w/o Noise</td>
<td>ImageNet</td>
<td>2</td>
<td>66.4</td>
</tr>
</tbody>
</table>

### 4.4 Ablation study

#### With vs without noise.** We compare the searched models with and without noises on two commonly used search spaces in Table 4. NoisyDARTS robustly escapes from the performance collapse across different search spaces and datasets. Note that without noise, the differentiable approach performs severely worse and obtains only 66.4% top-1 accuracy on the ImageNet classification task. In contrast, our simple yet effective method can find a state-of-the-art model with 76.1%.

#### Noise vs. Dropout ** Dropout [18] can be regarded as a type of special noise, which is originally designed to avoid overfitting. We use a special type of Dropout: DropPath [55] to act as a baseline, which is a drop-in replacement of our noise paradigm. We search on NAS-Bench-201 using different DropPath rates $r_{drop} \in \{0.1, 0.2\}$ and report the results in Table 5. It appears that Dropout produces much worse results.

#### Zero-mean (unbiased) noise vs. biased noise.** Experiments in Table 4 and 3 verify the necessity of the unbiased design, otherwise it brings in a deterministic bias. We can observe that the average performance of the searched models decreases while the bias increases. Eventually, it fails to overcome the collapse problem because larger biases overshoot the gradient and misguide the whole optimization process.
According to [51], the collapse is likely to occur when the maximal eigenvalue $\lambda_{\text{max}}^\alpha$ increases rapidly (whose Hessian matrix is calculated only on a snapshot of $\alpha$, i.e. single-point), under which condition some early stopping strategy was involved to avoid the collapse. To verify their claim, we search with DARTS and NoisyDARTS across 7 seeds and plot the calculated Hessian eigenvalues per epoch in Figure 1.

Remarkably, both DARTS and our method show a similar trend. We continue to train the supernet whilst eigenvalues keep increasing, but we still derive mostly good models with an average accuracy of 97.35%. It’s surprising to see that no obvious collapse occurs. Although the Hessian eigenvalue criterion benefits the elimination of bad models [51], it seems to mistakenly reject good ones. We also find the similar result on CIFAR-100, see Figure 3. We observe similar results in the reduced space too (see Section 3.10 in the supplementary).

We think that a single-point Hessian eigenvalue indicator at a local minimum cannot represent the curvatures of its wider neighborhood. It requires the wider landscape to be smoother to avoid the collapse. It is more clearly shown in Figure 2, where NoisyDARTS has a tent-like shape that eases the optimization.

Comparison with SDARTS. SDARTS [3], which performs perturbation on architectural weights to implicitly regularize the Hessian norm. However, we inject noise only into the skip connections’ output features or to all candidate operations, which suppresses the unfair advantage by disturbing the overly fluent gradient flow. Moreover, our method is efficient and nearly no extra cost is required. In contrast, SDARTS-ADV needs $2 \times$ search cost than ours.

Our method differs from SDARTS [8] in Hessian eigenvalue trend, as shown in Figure 1. SDARTS enjoys decreasing hessian eigenvalues while ours can have growing ones. The validation landscape of SDARTS is shown in Figure 2. SDARTS has a rather carpet-like landscape. It seems that too flat landscape of SDARTS may not correspond to a good model.

6 Conclusion

In this paper, we proposed a novel approach NoisyDARTS, to robustify differentiable architecture search. By injecting a proper amount of unbiased noise into candidate operations, we successfully let the optimizer be perceptible about the disturbed gradient flow. As a result, the unfair advantage is largely attenuated, and the derived models generally enjoy improved performance. Experiments show that NoisyDARTS can work effectively and robustly, regardless
of noise types. We achieved state-of-the-art results on several datasets and search spaces with low CO$_2$ emissions.

While most of the current approaches addressing the fatal collapse focus on designing various criteria to avoid stepping into the failure mode, our method stands out of the existing framework and no longer put hard limits as in [5, 28]. We review the whole optimization process to find out what leads to the collapse and directly control the unfair gradient flow, which is more fundamental than a stationary failure point analysis. We hope this would bring a novel insight for the NAS community to shift attention away from criteria-based algorithms.
References


