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].

1 Introduction

Differentiable architecture search [32] suffers from a well-known *performance collapse* issue noted by [6, 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 [6, 7, 25, 28, 31]. 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, 31] 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 [6, 28, 31] 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 [31] or too many skip connections in a cell [6, 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

This work was done when both authors were with Xiaomi.

[1] https://github.com/xiaomi-automl/NoisyDARTS

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 $3\times$ 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 α to denote its relative importance, and the outputs of all the paralleling choices are summed up using a softmax function σ. 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 [6] 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 [22] but still have difficulty navigating chaotic loss landscapes [26]. Combining it with a smoother loss function can relieve the pain for optimization which leads to better solutions [15, 26]. Previously, RobustDARTS [51] empirically finds that the collapse is highly related to the sharp curvature of the loss w.r.t α, 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 [17, 24, 33]. A recent study by [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,

$$\begin{aligned} &\alpha^* = \arg\min_{\alpha} \mathcal{L}_v(w,\alpha,z) = \arg\max_{\alpha} \mathbb{E}_{x,y\sim P_{val};z\sim P(z)} \log P(y|x,w^*,\alpha,z) \\ \text{s.t.} \quad &w^* = \arg\max_{w} \mathbb{E}_{x,y\sim P_{train};z\sim P(z)} \log P(y|x,w,\alpha,z) \end{aligned} \tag{1}$$

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 [32] for the rest notations. To incorporate noise and smooth DARTS (Equation 1), we propose a direct approach by setting,

$$z \sim N(\mu,\sigma). \tag{2}$$

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 μ and σ.

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 [6, 7, 28]. While being summed with other operations, a skip connection builds up a *residual structure* as in [16]. A similar form is also proposed in highway networks [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 [6, 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 μ We let $\tilde{x}$ be the noise injected into the skip operation, and α_s be the corresponding architectural weight. The loss of a skip connection operation can be written as,

$$\mathcal{L} = g(y), \quad y = f(\alpha_s) \cdot (x + \tilde{x}) \tag{3}$$

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

$$y^\star \approx f(\alpha_s) \cdot x \quad when \quad \tilde{x} \ll x. \tag{4}$$

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

$$\frac{\partial \mathcal{L}}{\partial \alpha_s} = \frac{\partial \mathcal{L}}{\partial y}\frac{\partial y}{\partial \alpha_s} = \frac{\partial \mathcal{L}}{\partial y}\frac{\partial f(\alpha_s)}{\partial \alpha_s}(x + \tilde{x}). \tag{5}$$

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,

$$\mathbb{E}_{\tilde{x}}[\nabla_{\alpha_s}] = \mathbb{E}_{\tilde{x}}\left[\frac{\partial \mathcal{L}}{\partial y}\frac{\partial f(\alpha_s)}{\partial \alpha_s}(x + \tilde{x})\right] \approx \frac{\partial \mathcal{L}}{\partial y^\star}\frac{\partial f(\alpha_s)}{\partial \alpha_s}(x + \mathbb{E}_{\tilde{x}}[\tilde{x}]). \tag{6}$$

Supposing that $\frac{\partial \mathcal{L}}{\partial y}$ is smooth, we can use $\frac{\partial \mathcal{L}}{\partial y^\star}$ to approximate the its small neighbor hood. Based on the premise stated in Equation 4, we take $\frac{\partial \mathcal{L}}{\partial y^\star}$ 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 σ The variance σ^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 σ=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 $\tilde{x} \sim \mathcal{N}(\mu, \sigma)$ 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 $\mathcal{O} = \{o^0_{i,j}, o^1_{i,j}, \cdots, o^{M-1}_{i,j}\}$ 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,

$$\overline{o}_{i,j}(x) = \sum_{k=1}^{M-1} f(\alpha_{o^k}) o^k(x) + f(\alpha_{o^{skip}}) o^{skip}(x + \tilde{x}). \tag{7}$$

The architecture search problem remains the same as the original DARTS, which is to alternately learn α^* and network weights w^* that minimize the validation loss $\mathcal{L}_{val}(\alpha^*, w^*)$. 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 [62]. Lastly, it smooths the loss landscape w.r.t α_s (NFA is similar). If we denote validation loss as $\mathcal{L}_v$, 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 $z \sim N(\mathbf{0}, \sigma^2\mathbf{I})$, the expectation of the loss over z can be approximated by

$$\begin{aligned}
\mathbb{E}_z\left[\mathcal{L}_v(w, \alpha_s, z)\right] &\approx \mathbb{E}_z[\mathcal{L}_v(w, \alpha_s, \mathbf{0}) + \nabla_z \mathcal{L}_v(w, \alpha_s, \mathbf{0})z + \frac{1}{2} z^T \nabla_z^2 \mathcal{L}_v(w, \alpha_s, \mathbf{0})z] \\
&= \mathcal{L}_v(w, \alpha_s, \mathbf{0})\mathbb{E}_z \mathbf{1} + \nabla_{z=\mathbf{0}} \mathcal{L}_v(w, \alpha_s, \mathbf{0})\mathbb{E}_z z + \mathbb{E}_z \frac{1}{2} z^T \nabla_z^2 \mathcal{L}_v(w, \alpha_s, \mathbf{0})z] \\
&= \mathcal{L}_v(w, \alpha_s, \mathbf{0}) + \frac{\sigma^2}{2} Tr\{\nabla_z^2 \mathcal{L}_v(w, \alpha_s, \mathbf{0})\} \\
&\approx \mathcal{L}_v(w, \alpha_s, \mathbf{0}) + \frac{\beta \sigma^2 \alpha_s^2}{2} Tr\{\nabla_{\alpha_s}^2 \mathcal{L}_v(w, \alpha_s, \mathbf{0})\} \\
\text{where} \quad & z \sim N(\mathbf{0}, \sigma^2\mathbf{I}), \beta = \mathbb{E}\frac{1}{o_{skip}(x)^T o_{skip}(x)}, \ \mathbf{I} \text{ is unit matrix.}
\end{aligned} \tag{8}$$

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 [32], MobileNetV2's search space as in [1], four harder spaces (from S_1 to S_4) from [51]. 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 3×3 and 5×5, dilation convolution 3×3 and 5×5).

MobileNetV2's search space (Benchmark 2) It is the same as that in ProxylessNAS [1]. We search proxylessly on ImageNet in this space. It uses the standard MobileNetV2's backbone architecture [39], 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 [51]. S_1 is a preoptimized search space with two operation per edge, see [51] for the detail. For each edge in the DAG, S_2 has only {3 × 3 SepConv, SkipConnect}, S_3 has {3 × 3 SepConv, SkipConnect, Zero (None)}, and S_4 has {3 × 3 SepConv, Noise}. We search on three datasets for each search space, which makes 12 benchmarks.

Models	Params (M)	×+ (M)	Top-1 Acc (%)
P-DARTS [5]	3.4	532†	97.49
PC-DARTS [49]	3.6	558†	97.43
GDAS [13]	3.4	519†	97.07
NoisyDARTS-a	3.3	534	**97.63**
NoisyDARTS-b	3.1	511	97.53
DARTS⋆[32]	3.3	528†	97.00±0.14
SNAS⋆[47]	2.8	422†	97.15±0.02
PR-DARTS⋆ [24]	3.4	-	97.19±0.08
P-DARTS [5]‡	3.3±0.21	540±34	97.19±0.14
PC-DARTS [49]‡	3.7±0.57	592±90	97.11±0.22
RDARTS [51]	-	-	97.05±0.21
DARTS- [6]	3.5±0.13	583±22	97.41±0.08
NoisyDARTS	3.1±0.22	502±38	97.35±0.23
MixNet-M* [42]	4.9	359	97.90
SCARLET-A [8]	5.4	364	98.05
EfficientNet B0* [41]	5.2	387	98.10
NoisyDARTS-A-t*	4.3	447	**98.28**

† Computed from the authors' code ‡ Re-run their code with 4 independent searches.
⋆ Averaged on the single **best** model trained for several times
* Transferring ImageNet-pretrained models to CIFAR-10

Models	×+ (M)	Params (M)	Top-1 (%)
MobileNetV2 [39]	585	6.9	74.7
NASNet-A [65]	564	5.3	74.0
AmoebaNet-A [38]	555	5.1	74.5
MdeNAS[53]	-	6.1	74.5
P-DARTS [5]††	577	5.1	74.9*
PC-DARTS [48]	597	5.3	75.8
DARTS [32]	574	4.7	73.3
GDAS [13]	581	5.3	74.0
MnasNet-92 [43]	388	3.9	74.79
Proxyless-R [1]	320†	4.0	74.6
NoisyDARTS-A	446	4.9	**76.1**
FairNAS-C ‡ [9]	321	4.4	74.7
FairDARTS-B [7]	541	4.8	75.1
MobileNetV3 [19]	219	5.4	75.2
EfficientNet B0 [41]	390	5.3	77.2
MixNet-M [42]	360	5.0	77.0
NoisyDARTS-A◇	449	5.5	**77.9**

†† Searched on CIFAR-100
◇ NoisyDARTS-A with SE and Swish enabled

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, 1×1 convolution, 3×3 convolution and 3×3 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 S_1, S_2, and S_3 (from [51]), where DARTS severely suffers from the collapse owing to an excessive number of skip connections. For S_4 where skip operations are not present, we apply NFA. We kept the same hyper-parameters as [51] 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.

Data	Space	DARTS	DARTS^{ADA}	DARTS^{ES}	Ours	RDARTS^{L2}	SDARTS^{RS}	SDARTS^{ADV}	Ours
C10	S_1	95.34±0.71	96.97±0.08	96.95±0.07	**97.05±0.18**	97.22	97.22	**97.27**	**97.27**
	S_2	95.58±0.40	96.41±0.31	96.59±0.14	**96.59±0.11**	96.69	96.67†	96.59†	**96.71**
	S_3	95.88±0.85	97.01±0.34	96.29±1.14	**97.42±0.08**	97.49	97.47	97.51	**97.53**
	S_4	93.05±0.18	96.11±0.67	95.83±0.21	**97.22±0.08**	96.44	97.07	97.13	**97.29**
C100	S_1	70.07±0.41	75.06±0.81	71.10±0.81	**77.89±0.88**	75.75	76.49	77.67	**78.83**
	S_2	71.25±0.92	73.12±1.11	75.32±1.43	**78.15±0.44**	77.76	77.72	**79.44**	78.82
	S_3	70.99±0.24	75.45±0.63	73.01±1.79	**79.48±0.59**	76.01	78.91	78.92	**79.93**
	S_4	75.23±1.51	76.34±0.90	76.10±2.01	**78.37±0.42**	78.06	78.54	78.75	**78.84**
SVHN	S_1	90.12±5.50	97.41±0.07	97.20±0.09	**97.44±0.06**	95.21	97.14†	**97.51**†	**97.51**
	S_2	96.31±0.12	97.21±0.22	97.32±0.18	**97.60±0.08**	97.49	97.61	97.65	**97.66**
	S_3	96.00±1.01	97.42±0.07	97.22±0.19	**97.58±0.06**	97.52	**97.64**	97.60	97.63
	S_4	97.10±0.02	97.48±0.06	97.45±0.15	**97.59±0.09**	97.50	97.54	97.58	**97.67**

Table 2: Comparison in the reduced spaces of RobustDARTS [51]. For NoisyDARTS, we use NFA for S_4 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 [20] and AutoAugment [10], 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 [14] in Table 3. Our method surpasses SETN [12] with a clear margin using 3 fewer search cost. This again proves NoisyDARTS to be a robust and powerful method. Learnable and decayed σ are used for ablation purposes (see Section 4.4).

Searching GCN on ModelNet10. We follow the same setting as SGAS [25] to search GCN networks on ModelNet10 [46] and evaluate them on ModelNet40. Our models (see Figure 20 in the supplementary) are on par with SGAS as reported in Table 7.

Method	Cost (hrs)	CIFAR-10		CIFAR-100		ImageNet16-120	
		valid	test	valid	test	valid	test
DARTS [33]	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
RSPS [27]	2.2	80.42±3.58	84.07±3.61	52.12±5.55	52.31±5.77	27.22±3.24	26.28±3.09
SETN [12]	9.5	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 [11]	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 [31]*	-	**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 [21]*	-	89.66±0.29	93.08±0.13	30.87±16.40	31.01±16.38	40.61±0.09	41.07±0.09
PCDARTS[15]*	-	89.96±0.15	93.41±0.30	67.12±0.39	67.48±0.89	40.83±0.08	41.31±0.22
NoisyDARTS	3.2	**90.26±0.22**	**93.49±0.25**	**71.36±0.21**	**71.55±0.51**	**42.47±0.00**	**42.34±0.06**

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. *: reported by [4]

Method	Type	Dataset	Benchmark	Acc (%)
NoisyDARTS	w/ Noise	CIFAR-10	1	97.35±0.23
DARTS	w/o Noise	CIFAR-10	1	96.62±0.23*
NoisyDARTS	w/ Noise	ImageNet	2	76.1
DARTS	w/o Noise	ImageNet	2	66.4

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

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.

r_{drop}	CIFAR-10		CIFAR-100		ImageNet-16	
	val	test	val	test	val	test
0.1	63.10±17.68	66.02±18.63	38.66±15.72	38.75±15.72	18.59±11.61	18.05±11.47
0.2	51.54±0.00	55.15±0.00	28.74±0.00	28.86±0.00	11.60±0.00	10.87±0.00
ours	**90.26±0.22**	**93.49±0.25**	**71.36±0.21**	**71.55±0.51**	**42.47±0.00**	**42.34±0.06**

Table 5: Applying Drop-path in replace of noise on NAS-Bench-201.

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.

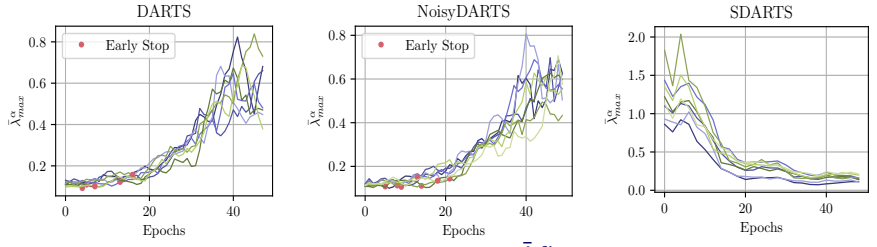


Figure 1: Smoothed maximal Hessian eigenvalues $\bar{\lambda}^{\alpha}_{max}$ [51] during the optimization on CIFAR-10. [51] suggests that the optimization should stop early at the marked points. SDARTS [3] regularizes $\bar{\lambda}^{\alpha}_{max}$ while searching. However, without doing so, we don't see the collapse in NoisyDARTS. DARTS has an average accuracy of 96.9% while we have 97.35%.

5 Discussion about the Single-point Hessian eigenvalue

According to [51], the collapse is likely to occur when the maximal eigenvalue λ^{α}_{max} increases rapidly (whose Hessian matrix is calculated only on a snapshot of α, 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.

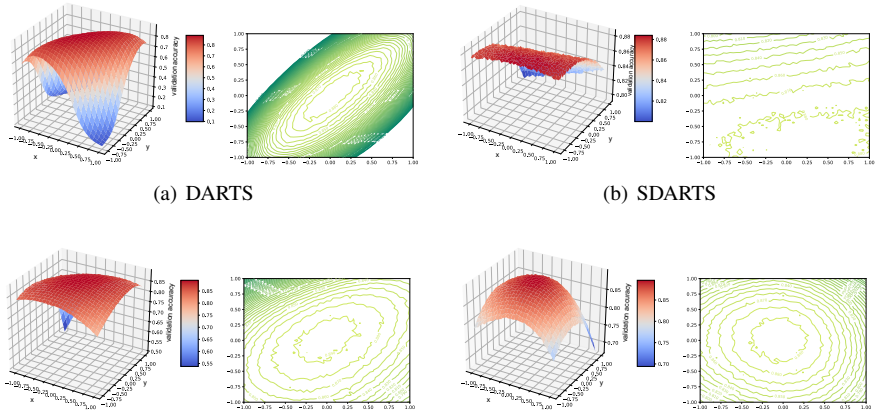
We observe similar results in the reduced space too (see Section 3.10 in the supplementary). **We think that a single-point Hessian eigenvalue indicator at a local minimum cannot represent the curvatures of its wider neighborhood. It requires the wider landscape be smoother to avoid the collapse**. It 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 [3] 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



(a) DARTS (b) SDARTS

(c) NoisyDARTS, NFA (d) NoisyDARTS, OFS

Figure 2: The landscape of validation accuracy w.r.t the architectural weights on CIFAR-10 and the corresponding contours. Following [3], axis x and y are orthogonal gradient direction of validation loss w.r.t. architectural parameters α, axis z refers to the validation accuracy. The related stand-alone model accuracies and Hessian eigenvalues are 96.96%/0.3388, 97.21%/0.1735, 97.42%/0.4495 respectively.

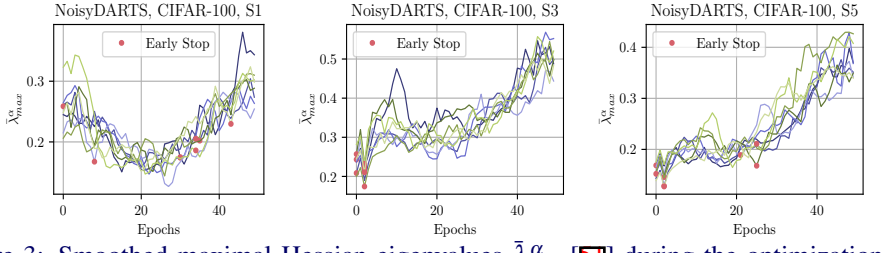


Figure 3: Smoothed maximal Hessian eigenvalues $\bar{\lambda}^{\alpha}_{max}$ [51] during the optimization on CIFAR-100 in reduced search spaces S_1, S_3, S_5 from [51]. We observe the similar growing trend. Notwithstanding, we achieve a state-of-the-art 16.28% test error rate in S_5.

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

[1] Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware. In *ICLR*, 2019.

[2] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

[3] Xiangning Chen and Cho-Jui Hsieh. Stabilizing Differentiable Architecture Search via Perturbation-based Regularization. In *ICML*, 2020.

[4] Xiangning Chen, Ruochen Wang, Minhao Cheng, Xiaocheng Tang, and Cho-Jui Hsieh. DrNAS: Dirichlet neural architecture search. In *ICLR*, 2021.

[5] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive Differentiable Architecture Search: Bridging the Depth Gap between Search and Evaluation. In *ICCV*, 2019.

[6] Xiangxiang Chu, Xiaoxing Wang, Bo Zhang, Shun Lu, Xiaolin Wei, and Junchi Yan. DARTS-: Robustly stepping out of performance collapse without indicators. In *ICLR*, 2020.

[7] Xiangxiang Chu, Tianbao Zhou, Bo Zhang, and Jixiang Li. Fair DARTS: Eliminating Unfair Advantages in Differentiable Architecture Search. *ECCV*, 2020.

[8] Xiangxiang Chu, Bo Zhang, Qingyuan Li, and Ruijun Xu. SCARLET-NAS: Bridging the Gap Between Stability and Fairness in Neural Architecture Search. In *ICCVW*, 2021.

[9] Xiangxiang Chu, Bo Zhang, and Ruijun Xu. FairNAS: Rethinking Evaluation Fairness of Weight Sharing Neural Architecture Search. In *ICCV*, 2021.

[10] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. AutoAugment: Learning Augmentation Policies from Data. In *CVPR*, 2019.

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, pages 248–255. IEEE, 2009.

[12] Xuanyi Dong and Yi Yang. One-shot neural architecture search via self-evaluated template network. In *ICCV*, pages 3681–3690, 2019.

[13] Xuanyi Dong and Yi Yang. Searching for a Robust Neural Architecture in Four GPU Hours. In *CVPR*, pages 1761–1770, 2019.

[14] Xuanyi Dong and Yi Yang. NAS-Bench-102: Extending the Scope of Reproducible Neural Architecture Search. In *ICLR*, 2020.

[15] Caglar Gulcehre, Marcin Moczulski, Francesco Visin, and Yoshua Bengio. Mollifying networks. In *ICLR*, 2017.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, pages 770–778, 2016.

[17] Zhezhi He, Adnan Siraj Rakin, and Deliang Fan. Parametric Noise Injection: Trainable Randomness to Improve Deep Neural Network Robustness Against Adversarial Attack. In *CVPR*, 2019.

[18] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[19] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for MobileNetV3. In *ICCV*, 2019.

[20] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-Excitation Networks. In *CVPR*, pages 7132–7141, 2018.

[21] Shoukang Hu, Sirui Xie, Hehui Zheng, Chunxiao Liu, Jianping Shi, Xunying Liu, and Dahua Lin. DSNAS: Direct Neural Architecture Search without Parameter Retraining. In *CVPR*, pages 12084–12092, 2020.

[22] Bobby Kleinberg, Yuanzhi Li, and Yang Yuan. An Alternative View: When Does SGD Escape Local Minima? In *ICML*, pages 2698–2707, 2018.

[23] Alex Krizhevsky, Geoffrey Hinton, et al. Learning Multiple Layers of Features from Tiny Images. Technical report, Citeseer, 2009.

[24] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 656–672. IEEE, 2019.

[25] Guohao Li, Guocheng Qian, Itzel C Delgadillo, Matthias Müller, Ali Thabet, and Bernard Ghanem. Sgas: Sequential greedy architecture search. In *CVPR*, 2020.

[26] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *NIPS*, pages 6389–6399, 2018.

[27] Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. In *Uncertainty in Artificial Intelligence*, pages 367–377. PMLR, 2020.

[28] 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.

[29] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *ECCV*, 2014.

[30] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal Loss for Dense Object Detection. In *ICCV*, pages 2980–2988, 2017.

[31] 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*, pages 19–34, 2018.

[32] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable Architecture Search. In *ICLR*, 2019.

[33] Xuanqing Liu, Minhao Cheng, Huan Zhang, and Cho-Jui Hsieh. Towards robust neural networks via random self-ensemble. In *ECCV*, pages 369–385, 2018.

[34] Niv Nayman, Asaf Noy, Tal Ridnik, Itamar Friedman, Rong Jin, and Lihi Zelnik-Manor. XNAS: Neural Architecture Search with Expert Advice. In *NeurIPS*, 2019.

[35] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPSW*, 2011.

[36] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*, pages 8024–8035, 2019.

[37] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient Neural Architecture Search via Parameter Sharing. In *ICML*, 2018.

[38] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *AAAI*, volume 33, pages 4780–4789, 2019.

[39] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *CVPR*, pages 4510–4520, 2018.

[40] Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. In *NIPS*, pages 2377–2385, 2015.

[41] Mingxing Tan and Quoc V Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *ICML*, 2019.

[42] Mingxing Tan and Quoc V. Le. MixConv: Mixed Depthwise Convolutional Kernels. In *BMVC*, 2019.

[43] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, and Quoc V Le. Mnasnet: Platform-Aware Neural Architecture Search for Mobile. In *CVPR*, 2019.

[44] 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.

[45] Dongxian Wu, Yisen Wang, and Shu-tao Xia. Revisiting Loss Landscape for Adversarial Robustness. *arXiv preprint arXiv:2004.05884*, 2020.

[46] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, pages 1912–1920, 2015.

[47] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. SNAS: Stochastic Neural Architecture Search. In *ICLR*, 2019.

[48] 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*, 2019.

[49] 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. URL https://openreview.net/forum?id=BJlS634tPr.

[50] Kaicheng Yu, Christian Sciuto, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. Evaluating the search phase of neural architecture search. In *ICLR*, 2020.

[51] Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. Understanding and robustifying differentiable architecture search. In *ICLR*, 2020. URL https://openreview.net/forum?id=H1gDNyrKDS.

[52] Yuchen Zhang, Percy Liang, and Moses Charikar. A Hitting Time Analysis of Stochastic Gradient Langevin Dynamics. In *Conference on Learning Theory*, pages 1980–2022, 2017.

[53] Xiawu Zheng, Rongrong Ji, Lang Tang, Baochang Zhang, Jianzhuang Liu, and Qi Tian. Multinomial Distribution Learning for Effective Neural Architecture Search. In *ICCV*, pages 1304–1313, 2019.

[54] Pan Zhou, Caiming Xiong, Richard Socher, and Steven Hoi. Theory-Inspired Path-Regularized Differential Network Architecture Search. In *NeurIPS*, 2020.

[55] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning Transferable Architectures for Scalable Image Recognition. In *CVPR*, volume 2, 2018.