NOISY DIFFERENTIABLE ARCHITECTURE SEARCH

Anonymous authors
Paper under double-blind review

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. Such an aggregation 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 allow fair competition for candidate operations. We name this novel approach as 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. Compared with the existing work, our method features extreme simplicity and acts as a new strong baseline. Experiments are performed across various search spaces, datasets, and tasks, where our method robustly achieves state-of-the-art results.

1 INTRODUCTION

Performance collapse from an excessive number of skip connections in the inferred model is a fatal drawback for differentiable architecture search (Liu et al., 2019), quite an amount of previous research has focused on addressing this issue (Chen et al., 2019b; Zela et al., 2020; Liang et al., 2019; Chu et al., 2020; Li et al., 2020). Among them, Chu et al. (2020) conclude that it is due to an unfair advantage in an exclusively competitive environment. Under this perspective, early-stopping methods like Zela et al. (2020); Liang et al. (2019) or greedy pruning (Li et al., 2020) 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 (Chen et al., 2019b; Liang et al., 2019; Zela et al., 2020) 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 Zela et al. (2020) or too many skip connections in a cell (Chen et al., 2019b; Liang et al., 2019)), 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.

Inspiringly, Chu et al. (2020) step out of this framework with an investigation of the collapse, which they attributed to two indispensable factors: *unfair advantage* under *competitive environment*. We build our work on top of their findings. Our contributions can be summarized in the following aspects.

- 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. While Chu et al. (2020) break the competition to prevent the one-sided unfair advantage from taking effect, we instead directly diminish such unfairness. This approach also has an effect of smoothing loss landscape w.r.t. architectural parameters α.
- 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 Hessian norm as an indicator of the collapse (Zela et al., 2020), since the narrow local Hessian norm of can't represent the curvatures of its wider neighborhood.

- Extensive experiments performed across various search spaces 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 cost than RDARTS (Zela et al., 2020). The source code and hyper-parameters are provided to reproduce our work.

2 Related work

Differentiable architecture search DARTS (Liu et al., 2019) has widely disseminated the paradigm of solving architecture search with gradient descent (Cai et al., 2019; Nayman et al., 2019; Xie et al., 2019). 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 very efficient, it is known to be unstable to reproduce (Yu et al., 2020).

Endeavors to improve the performance collapse in DARTS Several previous works have focused on addressing the collapse. For instance, Chen et al. (2019b) 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 Liang et al. (2019) simply early stop when there are exactly two skip connections in a cell. Zela et al. (2020) discover 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. There is a concurrent work (Chen & Hsieh, 2020), which undertake a perturbation-based approach on the architectural weights to implicitly regularize the Hessian norm, whose landscape might be too flat to traverse.

3 Noisy DARTS

3.1 Motivation

We are motivated by two distinct and orthogonal aspects: how to make the optimization easier and how to remove 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 (Kleinberg et al., 2018) but still have difficulty navigating chaotic loss landscapes (Li et al., 2018). Combining it with a smoother loss function can relieve the pain for optimization which leads to better solutions (Gulcehre et al., 2017; Li et al., 2018). Previously, Zela et al. (2020) empirically find 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 relative 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 Liu et al. (2018); He et al. (2019); Lecuyer et al. (2019). A recent study by Wu et al. (2020) 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,

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

where a random variable z is subject to the Dirac distribution and added to the intermediate features. For a multiplicative version, we can simply set $z \sim \delta(z-1)$.

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 C.1. 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 (Chen et al., 2019b; Liang et al., 2019; Chu et al., 2020). While being summed with other operations, a skip connection builds up a *residual structure* as in He et al. (2016). A similar form is also proposed in highway networks (Srivastava et al., 2015), see Figure 1. 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 (Chen et al., 2019b; Chu et al., 2020), it leads us to overestimate its relative importance, while others are underevaluated. **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 for NoisyDARTS.

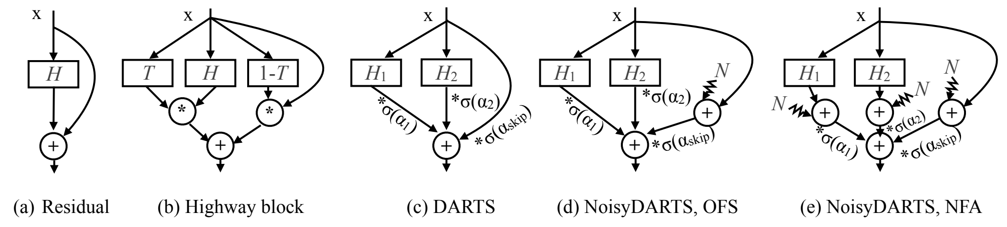


Figure 1: Various residual structures. H and T refer to non-linear units, N is the injected noise. A highway block learns how much it needs to transform (T) or to carry ($1-T$). DARTS decides the relative importance of operations between any two nodes by $\sigma(\alpha)$. Note here σ refers to softmax.

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 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_{skip} be the corresponding architectural weight. The loss of a skip connection operation can then be written as,

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

where $g(y)$ is the validation loss function and $f(\alpha_{skip})$ is the softmax operation for α_{skip}. Approximately, when the noise is much smaller than the output features, we have

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

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

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

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,

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

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 $\mathbb{E}\left[\tilde{x}\right]$ in the gradient of skip connection, to keep the gradient unbiased, $\mathbb{E}\left[\tilde{x}\right]$ 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,

Algorithm 1 NoisyDARTS-OFS (default and recommended)

```
1: Input: Architecture parameters $\alpha_{i,j}$, network weights $w$ , noise's standard variance $\sigma$, $Epoch_{max}$.
2: while not reach $Epoch_{max}$ do
3:     Inject random Gaussian noise $\tilde{x}$ into the skip connections' output.
4:     Update weights $w$ by $\nabla_w \mathcal{L}_{train}(w, \alpha)$
5:     Update architecture parameters $\alpha$ by $\nabla_\alpha \mathcal{L}_{val}(w, \alpha)$
6: end while
7: Derive the final architecture according to learned $\alpha$.
```

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

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. The NFA version is in Algorithm 2 (see B).

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 (Zhang et al., 2017). Lastly, it smooths the loss landscape w.r.t α_{skip} (NFA is similar). This role can be explained due to the fact that our approach implicitly controls the loss landscape, detailed by Equation 8. Its role can be better understood via the visualization in Figure 4, where DARTS obtains a sharp landscape with oval contours. In contrast, our smoothed version has round ones.

$$\begin{aligned}
\mathbb{E}_{z\sim N(0,\sigma^2)}\left[L_{valid}(w,\alpha_{skip},z)\right] &\approx \mathbb{E}_{z\sim N(0,\sigma^2)}[L_{valid}(w,\alpha_{skip},\mathbf{0}) + \nabla_z L_{valid}(w,\alpha_{skip},\mathbf{0})z \\
&\quad + \frac{1}{2}z^T\nabla_z^2 L_{valid}(w,\alpha_{skip},\mathbf{0})z] \\
&= L_{valid}(w,\alpha_{skip},\mathbf{0}) + \frac{\sigma^2}{2}Tr\{\nabla_z^2 L_{valid}(w,\alpha_{skip},\mathbf{0})\} \\
&\approx L_{valid}(w,\alpha_{skip},\mathbf{0}) + \frac{\beta\sigma^2\alpha_{skip}^2}{2}Tr\{\nabla_{\alpha_{skip}}^2 L_{valid}(w,\alpha_{skip},\mathbf{0})\} \\
\text{where} \qquad &\beta = \mathbb{E}\frac{1}{o_{skip}(x)^T o_{skip}(x)}
\end{aligned} \tag{8}$$

4 Experiments

Search spaces and Datasets To verify the validity of our method, we adopt several search spaces: the DARTS search space from Liu et al. (2019), MobileNetV2's search space as in Cai et al. (2019), four harder spaces (from S_1 to S_4) from Zela et al. (2020). We use NAS-Bench-201 (Dong & Yang, 2020) to benchmark our methods. As for datasets, we use CIFAR-10, CIFAR-100 (Krizhevsky et al., 2009), SVHN (Netzer et al., 2011) and ImageNet (Deng et al., 2009). Details are provided in section C.4. We also search for GCNs on ModelNet (Wu et al., 2015) as in Li et al. (2020) (see C.7).

4.1 Searching Results

Searching on CIFAR-10. In the search phase, we use similar hyperparameters and tricks as Liu et al. (2019). All experiments are done on a Tesla V100 with PyTorch 1.0 (Paszke et al., 2019). The search phase takes about 0.4 GPU day. 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 5 (see C). For training a single model, we use the same strategy and data processing tricks as Chen et al. (2019b); Liu et al. (2019), 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, which is shown in Figure 11 and Table 11 (see D).

Searching in the reduced search spaces of RobustDARTS. We also study the performance of our approach under reduced search spaces. Particularly we use OFS for S_1, S_2, and S_3 (in Zela et al. (2020)), 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 exact same hyper-parameters as Zela et al. (2020) for training every single model to make a fair comparison. Since the unfair advantage is intensified in the reduced search spaces, we use a stronger Gaussian noise. As before, we don't utilize any regularization tricks. The results are given in Table 2 and Table 13 (see C.11). Each search is repeated only three times to obtain the average.

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 C.5). As for training searched models, we use similar training tricks as EfficientNet (Tan & Le, 2019). The evolution of dominating operations during the search is illustrated in Figure 8. Compared with DARTS (66.4%), the injected noise in NoisyDARTS successfully eliminates the unfair advantage. Our model NoisyDARTS-A (see Figure 6 in C.11) 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 (Hu et al., 2018) and AutoAugment (Cubuk et al., 2019), 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 (Dong & Yang, 2020) in Table 3. Our method (fixed σ) surpasses SETN (Dong & Yang, 2019a) 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).

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 (see Figure 11 and 12). NoisyDARTS-A (see Figure 6) is searched on ImageNet in the MobileNetV2-like search space as in Wu et al. (2019).

Models	Params (M)	$\times +$ (M)	Top-1 Acc (%)	Cost (G·d)
NASNet-A (2018)	3.3	608[†]	97.35	2000
ENAS (2018)	4.6	626[†]	97.11	0.5
DARTS[⋆](2019)	3.3	528[†]	97.00±0.14	0.4
SNAS[⋆](2019)	2.8	422[†]	97.15±0.02	1.5
GDAS (2019b)	3.37	519[†]	97.07	0.3
P-DARTS (2019b)	3.4	532[†]	97.5	0.3
P-DARTS (2019b)[‡]	3.3±0.21	540±34	97.19±0.14	0.3
PC-DARTS (2020)[‡]	3.68±0.57	592±90	97.11±0.22	0.1
RDARTS (2020)	-	-	97.05±0.21	1.6
SDARTS-RS (2020)	3.4	-	97.33±0.03	0.4
NoisyDARTS	3.06±0.22	502±38	97.30±0.23	0.4
NoisyDARTS-a	3.25	534	97.63	0.4
NoisyDARTS-b	3.09	511	97.53	0.4
MixNet-M[*] (2019)	4.9	359	97.90	≈3k
EB0[*] (2019)	5.2	387	98.10	≈3k
NoisyDARTS-A-t[*]	4.3	447	98.28	12

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

Models	$\times +$ (M)	Params (M)	Top-1 (%)	Cost (G · d)
MobileNetV2 (2018)	585	6.9	74.7	-
NASNet-A (2018)	564	5.3	74.0	1800
AmoebaNet-A (2019)	555	5.1	74.5	≈3k
MnasNet-92 (2019)	388	3.9	74.79	≈4k
MdeNAS(2019)	-	6.1	74.5	0.16
DARTS (2019)	574	4.7	73.3	0.5
GDAS (2019b)	581	5.3	74.0	0.2
FBNet-C (2019)	375	5.5	74.9	9
Proxyless-R (2019)	320[†]	4.0	74.6	8.3
P-DARTS (2019b)[††]	577	5.1	74.9[*]	0.3
PC-DARTS (2019)	597	5.3	75.8	3.8
NoisyDARTS-A	446	4.9	76.1	12
MobileNetV3 (2019)	219	5.4	75.2	≈3k
MixNet-M (2019)	360	5.0	77.0	≈3k
NoisyDARTS-A[◇]	449	5.5	77.9	12

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

Table 2: Comparison in the reduced spaces of RobustDARTS. For NoisyDARTS, we use NFA for S_4, OFS for the rest. [†]: 16 initial channels (retrained). Three right columns are the best out of three runs.

Data	Space	DARTS	DARTS-ADA	DARTS-ES	NoisyDARTS	RDARTS (L2)	SDARTS-RS	SDARTS-ADV	NoisyDARTS
C-10	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**
C-100	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**

4.2 Transferred Results

Transferred results on object detection. We further evaluate the transferability of our searched models on the COCO objection task (Lin et al., 2014). Particularly, we utilize a drop-in replacement for the backbone based on Retina (Lin et al., 2017). As shown in Table 9, our model obtains the best transferability than other models under the mobile settings. Detailed setting is provided in section C.3.

Transferring ImageNet models to CIFAR-10. We transferred our NoisyDARTS-A model searched on ImageNet to CIFAR-10. Specifically, the transferred model NoisyDARTS-A-t achieved 98.28% top-1 accuracy with only 447M FLOPS, as shown in Table 1. Training details are listed in section C.2.

4.3 Relations to Other Work

Comparison with PNI. PNI (He et al., 2019) uses parametric noise to boost adversarial training. In contrast, we inject the fixed noise at the output of candidate operations and smooth the loss

Table 3: Comparison on NAS-Bench-201. Averaged on 3 searches. The best for DARTS-based methods is in bold and underlined, while the second best is in bold. *: reported by Chen et al. (2020)

Method	Cost (hrs)	CIFAR-10 valid	CIFAR-10 test	CIFAR-100 valid	CIFAR-100 test	ImageNet16-120 valid	ImageNet16-120 test
Random (2012)	0	90.93±0.36	93.70±0.36	70.60±1.37	70.65±1.38	42.92±2.00	42.96±2.15
DARTS-V1 (2019)	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 (2020)	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 (2019a)	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 (2019b)	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 (2019)*	-	**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 (2020)*	-	89.66±0.29	93.08±0.13	30.87±16.40	31.01±16.38	40.61±0.09	41.07±0.09
PC-DARTS (2020)*	-	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 (learnable σ)	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
NoisyDARTS (decayed σ)	3.2	87.00±0.00	90.59±0.00	66.35±0.00	67.34±0.00	40.75±0.00	40.08±0.00
NoisyDARTS (fixed σ)	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**

landscape of the bi-level search to avoid collapse. Moreover, parametric noise leads to collapse on NAS-Bench-201 (see learnable σ in Table 3).

Comparison with SDARTS. The concurrent work SDARTS (Chen & Hsieh, 2020), 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.

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%.

Table 4: **Left** : NoisyDARTS is robust across S_0 on CIFAR-10 and S_1 on ImageNet. *: Reported by Yu et al. (2020) **Right**: Ablation on additive Gaussian noise on CIFAR-10 (each search is run 8 times)

Type	Dataset	Acc (%)
w/ Noise	CIFAR-10	97.30±0.23
w/o Noise	CIFAR-10	96.62±0.23*
w/ Noise	ImageNet	76.1
w/o Noise	ImageNet	66.4

Type	μ	σ	Acc (%)	μ	σ	Acc (%)
Gaussian	-1.0	0.1	96.92±0.36	-1.0	0.2	97.14±0.15
Gaussian	-0.5	0.1	97.13±0.20	-0.5	0.2	97.07±0.12
Gaussian	0.0	0.1	**97.21±0.21**	0.0	0.2	**97.30±0.23**
Gaussian	0.5	0.1	97.02±0.21	0.5	0.2	97.16±0.15
Gaussian	1.0	0.1	96.89±0.26	1.0	0.2	96.82±0.57

Zero-mean (unbiased) noise vs. biased noise. Experiments in Table 4 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. The genotypes of these searched cells are detailed in Table 12 (see C.9).

Noise vs. Dropout Dropout (Hinton et al., 2012) can be regarded as a type of special noise, which is designed by human expert to avoid overfitting. We use a special type of Dropout: DropPath (Zoph et al., 2018) 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 $r_{drop} \in \{0.1, 0.2, 0.3, 0.4\}$ and report the results in Table 5. We attribute its failure to the limited regularization of the unfair gradient.

Table 5: Applying Drop-path in replace of noise on NAS-Bench-201. C: CIFAR, I: ImageNet

r_{drop}	C10 test	C100 test	I16 test
0.1	66.02±18.63	38.75±15.72	18.05±11.47
0.2	55.15±0.00	28.86±0.00	10.87±0.00
0.3	55.15±0.00	28.86±0.00	10.87±0.00
0.4	55.15±0.00	28.86±0.00	10.87±0.00

Table 6: NFA, ES, OFS on NAS-Bench-201.

Method	C10 test	C100 test	I16 test
NFA, σ=0.2	54.30±0.00	15.61±0.00	16.32±0.00
NFA, σ=0.3	83.04±4.28	52.22±7.44	28.40±3.65
NFA, σ=0.4	91.60±1.74	68.26±1.59	41.57±2.59
ES, σ=0.2	54.30±0.00	15.61±0.00	16.32±0.00
ES, σ=0.3	54.30±0.00	15.61±0.00	16.32±0.00
ES, σ=0.4	59.84±9.59	23.39±13.48	17.01±1.20
OFS	**93.49±0.25**	**71.55±0.51**	**42.34±0.06**

Noise For All (NFA) vs. Only For Skip connection (OFS) Applying noise to skip connections is not an ad-hoc decision. In theory, we can interfere with the optimization by injecting the noise to any operation. The underlying philosophy is that only those operations that can work robustly against noises will win the race without unfair advantage. We compare the settings of NFA, ES (noise for all but excluding skip), OFS in Table 6. *This proves noise injection to skip connection is critical for better searching performance.* Note that this approach also obtains much better result than DARTS. However, it requires a bit of trial-and-error to control the σ when there are many candidate operations. Therefore, if otherwise specified, we use OFS as the default choice throughout the paper.

Decayed σ vs fixed σ Apart from fixed σ setting, we can also gradually decay σ to zero as a feasible solution, which is also easy to implement. In this annealing approach, exploration is encouraged at the early stage to avoid local minimum, then after reaching a relative smooth landscape, it slows down to find a good solution. However, decayed σ is worse than fixed version, shown in Table 3.

Learnable σ vs fixed σ We further make σ learnable. We use the reparametrization trick to update σ by back-propagation. The results are shown in Table 3. It turns out that σ **decreases rapidly during the searching process**, which cannot regularize the unfair advantage and results in the collapse.

More ablation studies about the noise types and the injection methods are provided in C.1.

5 HESSIAN EIGENVALUE IS NOT A PERFECT INDICATOR OF COLLAPSE

We search across 7 seeds and plot the calculated values in Fig. 3. Remarkably, both DARTS and our method show a similar trend. According to Zela et al. (2020), the collapse occurs when the maximal eigenvalue increases rapidly (above a given threshold). Under such condition, early stopping is required to avoid the collapse. In contrast, we continue to train the supernet whilst eigenvalues keep increasing, but we still derive mostly good models with an average accuracy of 97.30%. It's surprising to see that no obvious collapse occurs in our case. Although the Hessian eigenvalue criterion benefits the elimination of bad models (Zela et al., 2020), it seems to mistakenly reject good ones (Figure 2).

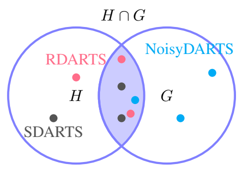


Figure 2: Exemplary illustration on the relation of the set (denoted as H) of models found with low Hessian norms and the set (G) of models with better test accuracy. The Hessian norm criterion tends to reject a part of good models.

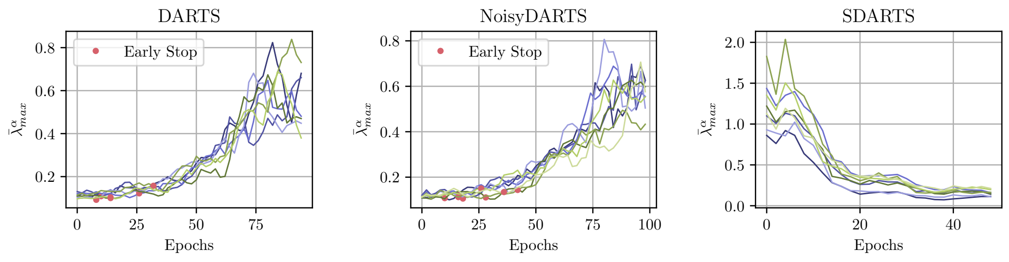
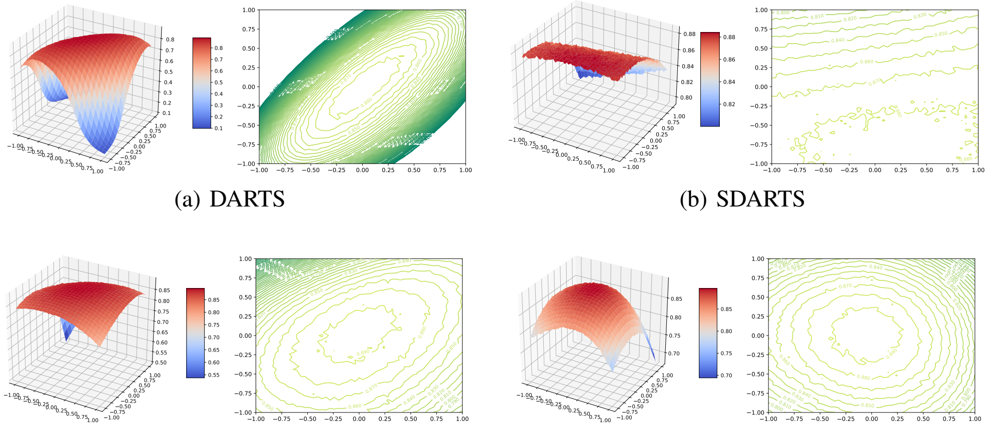


Figure 3: Smoothed maximal Hessian eigenvalues $\bar{\lambda}^{\alpha}_{max}$ (Zela et al., 2020) during the optimization on CIFAR-10. Zela et al. (2020) suggest that the optimization should stop early at the marked points. Without doing so, we don't see the collapse in NoisyDARTS either. DARTS's models have an average accuracy of 96.9% while ours and SDARTS 97.3%.

We observe similar result in the reduced space too (see C.10). **It seems that a smoother landscape helps to avoid the collapse, not the Hessian eigenvalue indicator. Hessian norm at a local minimum cannot represent the curvatures of its wider neighborhood.** It is more clearly shown in Figure 4, where NoisyDARTS has a tent-like shape and SDARTS instead has a rather carpet-like landscape. It seems that too flat landscape of SDARTS may not correspond to a good model.



(a) DARTS (b) SDARTS

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

Figure 4: The landscape of validation accuracy w.r.t the architectural weights on CIFAR-10 and the corresponding contours. Following Chen & Hsieh (2020), 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 accuracy and Hessian eigenvalues are 96.96%/0.3388, 97.21%/0.1735, 97.42%/0.4495 respectively.

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 Chen et al. (2019b); Liang et al. (2019). 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 their attention away from criteria-based algorithms.

REFERENCES

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(1):281–305, 2012.

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

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*, 2019a.

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

Xiangning Chen, Ruochen Wang, Minhao Cheng, Xiaocheng Tang, and Cho-Jui Hsieh. Drnas: Dirichlet neural architecture search. *arXiv preprint arXiv:2006.10355*, 2020.

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

Xiangxiang Chu, Tianbao Zhou, Bo Zhang, and Jixiang Li. Fair darts: Eliminating unfair advantages in differentiable architecture search. *ECCV*, 2020.

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

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

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

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

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

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

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

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.

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.

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.

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

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

Bobby Kleinberg, Yuanzhi Li, and Yang Yuan. An alternative view: When does sgd escape local minima? In *International Conference on Machine Learning*, pp. 2698–2707, 2018.

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

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)*, pp. 656–672. IEEE, 2019.

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

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, pp. 6389–6399, 2018.

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

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.

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.

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

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

Xuanqing Liu, Minhao Cheng, Huan Zhang, and Cho-Jui Hsieh. Towards robust neural networks via random self-ensemble. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 369–385, 2018.

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

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.

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*, pp. 8024–8035, 2019.

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

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

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

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

Dimitrios Stamoulis, Ruizhou Ding, Di Wang, Dimitrios Lymberopoulos, Bodhi Priyantha, Jie Liu, and Diana Marculescu. Single-Path NAS: Designing Hardware-Efficient ConvNets in less than 4 Hours. *ECML PKDD*, 2019.

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

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

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

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.

Dongxian Wu, Yisen Wang, and Shu-tao Xia. Revisiting loss landscape for adversarial robustness. *arXiv preprint arXiv:2004.05884*, 2020.

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*, pp. 1912–1920, 2015.

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

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 *International Conference on Learning Representations*, 2019.

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`.

Kaicheng Yu, Christian Sciuto, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. Evaluating the search phase of neural architecture search. In *ICLR*, 2020. URL `https://openreview.net/forum?id=H1loF2NFwr`.

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`.

Yuchen Zhang, Percy Liang, and Moses Charikar. A hitting time analysis of stochastic gradient langevin dynamics. In *Conference on Learning Theory*, pp. 1980–2022, 2017.

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

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

A Analysis of multiplicative noise

We set the output of skip connection with multiplicative noise is $x' = x \cdot \tilde{x}$, where $\tilde{x}$ is sampled from a certain distribution. Similar to Section 3.2, the expectation of the gradient under multiplicative noise can be written as:

$$\mathbb{E}\left[\nabla_{skip}\right] = \mathbb{E}\left[\frac{\partial \mathcal{L}}{\partial y}\frac{\partial f(\alpha^{skip})}{\partial \alpha^{skip}}\left(x'\right)\right] \approx \frac{\partial \mathcal{L}}{\partial y^{\star}}\frac{\partial f(\alpha^{skip})}{\partial \alpha^{skip}}\left(x \cdot \mathbb{E}\left[\tilde{x}\right]\right). \tag{9}$$

Again notice that taking $\frac{\partial \mathcal{L}}{\partial y^{\star}}$ out of the expectation in Equation 9 requires Equation 4 be satisfied. To keep the gradient unbiased, $\tilde{x}$ should be close to 1. Thus, we use Gaussian distribution $\tilde{x} \sim \mathcal{N}(1, \sigma^2)$.

B Algorithm

Algorithm 2 NoisyDARTS-NFA

```
1: Input: Architecture parameters $\alpha_{i,j}$, network weights $w$ , noise's standard variance $\sigma$,
   $Epoch_{max}$.
2: while not reach $Epoch_{max}$ do
3:     Inject random Gaussian noise $\tilde{x}$ into all candidate operations' output.
4:     Update weights $w$ by $\nabla_w \mathcal{L}_{train}(w, \alpha)$
5:     Update architecture parameters $\alpha$ by $\nabla_\alpha \mathcal{L}_{val}(w, \alpha)$
6: end while
7: Derive the final architecture according to learned $\alpha$.
```

C More Experiments and Details

C.1 More Ablation Studies

Gaussian noise vs. uniform noise According to the analysis of Section 3.2, unbiased Gaussian noise is an appropriate choice that satisfies Equation 6. In the same vein, unbiased uniform noise should be equally useful. We compare both types of noise in terms of effectiveness in Table 7. Both have improved performance while Gaussian is slightly better. This can be loosely explained. As the output feature x from each skip connection tends to be Gaussian, i.e. $x \sim \mathcal{N}(\mu_1, \sigma_1^2)$ (see Fig. 7), a Gaussian noise $\tilde{x} \sim \mathcal{N}(0, \sigma_2^2)$ is preferred since the additive result shares the similar statistics, i.e., $x + \tilde{x} \sim N(\mu_1, \sigma_1^2 + \sigma_2^2)$.

Additive noise vs. multiplicative noise Apart from additive noise, we also blend the noise ($\mu = 1$) by multiplying it with the output x of skip connections, which is approximately effective as additive noise, see Table 7. We give a theoretical proof in Appendix A, similar to the one in Section 3.2. In general, we also notice that searching with the biased noise also outperforms DARTS. This could be empirically interpreted as that resolving the aggregation of skip connections is more critical, while a slight deviation during the optimization matters less.

Table 7: Experiments on different types of noise and mixing operations. Each search is run 8 times

Noise Type	μ	σ	Avg. Top-1 (%)
w/o Noise	-	-	97.00±0.14
Gaussian	0.0	0.1	97.21±0.21
Uniform	0.0	0.1	97.12±0.15
Gaussian	0.0	0.2	97.30±0.23
Uniform	0.0	0.2	97.15±0.23

Noise Mixture	μ	σ	Top-1 (%)
w/o Noise	-	-	97.00±0.14
Additive	0.0	0.1	97.21±0.21
Multiplicative	1.0	0.1	97.15±0.23
Additive	0.0	0.2	97.30±0.23
Multiplicative	1.0	0.2	97.22±0.23

Remove Skip Connection from the search space Skip connections are a necessary component but they are troublesome for DARTS. We remove this operation from the NAS-Bench-201 search space to study how well DARTS performs. Table 8 hints that DARTS can find relatively competitive architectures (no longer suffering performance collapse), but not as good as those found by state-of-the-art methods in Table 3, for instance, it has a CIFAR-10 test accuracy 88.98% vs. NoisyDARTS's 93.49%. We suggest that skip connections play an indispensable role in neural architecture search and have to be carefully dealt with as we did in NoisyDARTS.

Table 8: Removing skip connection from search space on NAS-Bench-201. C: CIFAR, I: ImageNet

Setting	C10 val	C10 test	C100 val	C100 test	I16 val	I16 test
w/ skip (DARTS)	39.77±0.00	54.30±0.00	15.03±0.00	15.61±0.00	16.43±0.00	16.32±0.00
w/o skip (DARTS)	85.67±1.30	88.98±0.85	63.17±1.00	62.82±1.74	33.74±2.69	33.29±2.66
NoisyDARTS	**90.26±0.22**	**93.49±0.25**	**71.36±0.21**	**71.55±0.51**	**42.47±0.00**	**42.34±0.06**

C.2 Training Setting on Transferred Results on CIFAR-10

The model is trained for 200 epochs with a batch size of 256 and a learning rate of 0.05. We set the weight decay to be 0.0, a dropout rate of 0.1 and a drop connect rate of 0.1. In addition, we also use AutoAugment as (Tan & Le., 2019).

C.3 Training Settings on COCO Object Detection

We use the MMDetection tool box since it provides a good implementation for various detection algorithms (Chen et al., 2019a). Following the same training setting as Lin et al. (2017), all models in Table 9 are trained and evaluated on the COCO dataset for 12 epochs. The learning rate is initialized as 0.01 and decayed by 0.1× at epoch 8 and 11.

Table 9: Object detection of various drop-in backbones.

Backbones	Params (M)	Acc (%)	AP (%)	AP_{50} (%)	AP_{75} (%)	AP_S (%)	AP_M (%)	AP_L (%)
MobileNetV2 (2018)	3.4	72.0	28.3	46.7	29.3	14.8	30.7	38.1
SingPath NAS (2019)	4.3	75.0	30.7	49.8	32.2	15.4	33.9	41.6
MnasNet-A2 (2019)	4.8	75.6	30.5	50.2	32.0	16.6	34.1	41.1
MobileNetV3 (2019)	5.4	75.2	29.9	49.3	30.8	14.9	33.3	41.1
MixNet-M (2019)	5.0	77.0	31.3	51.7	32.4	17.0	35.0	41.9
NoisyDARTS-A	5.5	77.9	33.1	53.4	34.8	18.5	36.6	44.4

C.4 Search spaces and 15 benchmarks

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 (Cai et al., 2019). We search proxylessly on ImageNet in this space. It uses the standard MobileNetV2's backbone architecture (Sandler et al., 2018), 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 6 for reference.

S_1-S_4 (Benchmark 3-14) These are reduced search spaces introduced by R-DARTS (Zela et al., 2020). S_1 is a preoptimized search space with two operation per edge, see Zela et al. (2020) 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.

NAS-Bench-201 (Benchmark 15) NAS-Bench-201 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).

C.5 Detailed Settings on ImageNet

We split the original training set into two datasets with equal capacity to act as our training and validation dataset. The original validation set is treated as the test set. We use the SGD optimizer with a batch size of 768. The learning rate for the network weights is initialized as 0.045 and it decays to 0 within 30 epochs following the cosine decay strategy. Besides, we utilize Adam optimizer ($\beta_1 = 0.5, \beta_2 = 0.999$) and a constant learning rate of 0.001.

C.6 Detailed Settings on NAS-Bench-201

For NAS-Bench-201 experiments, we adapt the code from Dong & Yang (2020). We only use the first-order DARTS optimization. We track the running statistics for batch normalization to be the same as DARTS (Liu et al., 2019). Each setting is run 3 times to obtain the average. We use a noise of $\sigma = 0.8$ regarding this particular search space.

C.7 Searching GCN Architectures on ModelNet10

We follow the same setting as SGAS (Li et al., 2020) to search GCN networks on ModelNet10 (Wu et al., 2015) and evaluate them on ModelNet40. Our models (see Figure 23) are on par with the greedy approach SGAS as reported in Table 10.

Table 10: 3D Object classification on ModelNet40. OA: overall accuracy

Methods	Params (M)	OA (%)
SGAS (Cri. 1 avg.)	8.78	92.69±0.20
SGAS (Cri. 1 best)	8.63	92.87
NoisyDARTS ($\sigma = 0.2$ avg.)	8.72	92.65±0.17
NoisyDARTS ($\sigma = 0.2$ best)	8.48	92.90
NoisyDARTS ($\sigma = 0.3$ avg.)	8.68	92.85±0.36
NoisyDARTS ($\sigma = 0.3$ best)	8.33	93.11
NoisyDARTS ($\sigma = 0.4$ avg.)	8.68	92.70±0.43
NoisyDARTS ($\sigma = 0.4$ best)	8.93	93.23

C.8 Evolution of NoisyDARTS architectural parameters

We plot the evolution of architectural parameters during the NoisyDARTS optimization in Fig. 5. The injected noise is zero-mean Gaussian with $\sigma = 0.2$. As normal cells are the main building blocks (18 out of 20) of the network, we see that the number of skip connections is much reduced. Compared with Liang et al. (2019) and Chen et al. (2019b), we don't set any hard limits for it. We also don't compute expensive Hessian eigenspectrum (Zela et al. (2020)) as a type of regularization for skip connections. Neither do we use Scheduled DropPath (Zoph et al. (2018)) or fixed drop-path during the searching. It confirms that by simply disturbing the gradient flow of skip connections, the unfair advantage is much weakened so that the optimization is fairer to deliver better performing models.

C.9 NoisyDARTS CIFAR-10 architectures with unbiased and biased noise

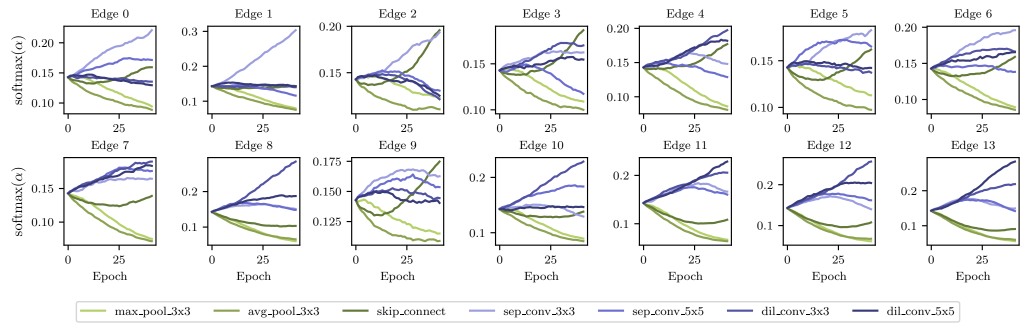
We give the details of best architectures when searching with unbiased noise in Table 11, and the biased ones in Table 12.

Table 11: Best NoisyDARTS architecture genotypes searched on CIFAR-10 with unbiased noise

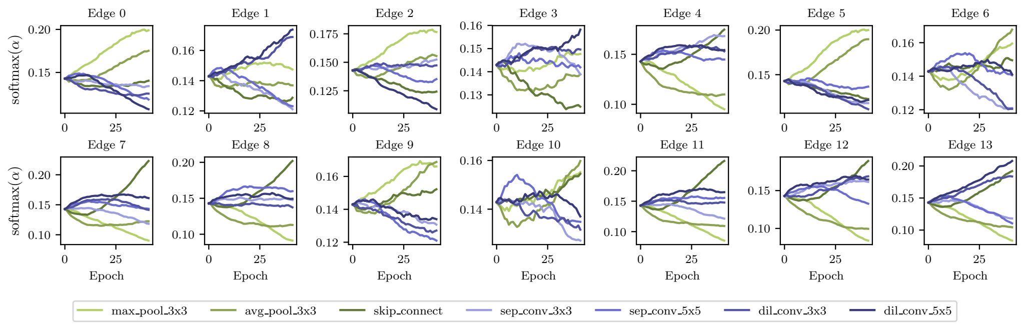
Model	Architecture Genotype
NoisyDARTS-a	Genotype(normal=[('sep_conv_3x3', 1), ('sep_conv_3x3', 0), ('skip_connect', 0), ('dil_conv_3x3', 2), ('dil_conv_3x3', 3), ('sep_conv_3x3', 1), ('dil_conv_5x5', 4), ('dil_conv_3x3', 3)], normal_concat=range(2, 6), reduce=[('max_pool_3x3', 0), ('dil_conv_5x5', 1), ('skip_connect', 2), ('max_pool_3x3', 0), ('skip_connect', 2), ('skip_connect', 3), ('skip_connect', 2), ('dil_conv_5x5', 4)], reduce_concat=range(2, 6))
NoisyDARTS-b	Genotype(normal=[('sep_conv_3x3', 1), ('sep_conv_3x3', 0), ('skip_connect', 2), ('sep_conv_3x3', 0), ('dil_conv_3x3', 3), ('skip_connect', 0), ('dil_conv_3x3', 3), ('dil_conv_3x3', 4)], normal_concat=range(2, 6), reduce=[('max_pool_3x3', 0), ('skip_connect', 1), ('max_pool_3x3', 0), ('skip_connect', 2), ('skip_connect', 2), ('max_pool_3x3', 0), ('skip_connect', 2), ('avg_pool_3x3', 0)], reduce_concat=range(2, 6))

C.10 More discussions about Hessian Indicator in Reduced Search Space

Specifically, when training these models from supposed early-stop points, we only obtain a lower average performance 97.02±0.21. How about other search spaces? We further evaluate the Hessian



(a) Normal cell



(b) Reduction cell

Figure 5: Evolution of architectural weights during the NoisyDARTS searching phase on CIFAR-10. Skip connections in normal cells are largely suppressed.

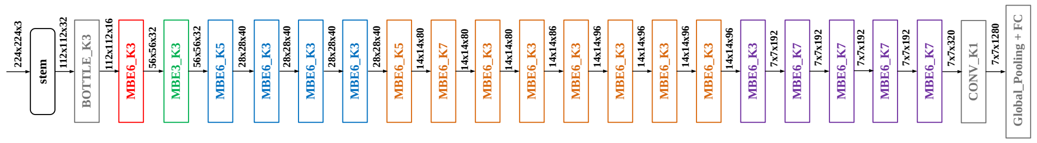


Figure 6: NoisyDARTS-A searched on ImageNet. Colors represent different stages.

eigenvalue trajectories of our method in the reduced search space, which are shown in Figure 9. When we search with injected Gaussian noise, we still observe an obvious growth of eigenvalues in both of two spaces. However, when being trained from scratch, the models derived from the last epoch (without early-stopping or any regularization tricks) perform much better than their proposed adaptive eigenvalue regularization method DARTS-ADA (Zela et al., 2020). Compared with their best effort L2 regularization (Zela et al., 2020) and another method SDARTS (Chen & Hsieh, 2020) based on implicit regularization of Hessian norm, we also have better performance in S_2 and comparable performance in S_3 (see Table 2). Notice we only use **3x fewer** searching cost. This reassures our observation that the Hessian norm (Zela et al., 2020) may not be an ideal indicator of performance collapse, because it rejects good models by mistake, as illustrated in Figure 2.

C.11 More details on reduced RobustDARTS experiments

Like on CIFAR-10, we repeatedly find the Hessian eigenvalues are both growing when searching with NoisyDARTS on CIFAR-100 and SVHN datasets (see Fig. 10), but models derived from these searching runs still outperform or are comparable to those from regularized methods like RDARTS (Zela et al., 2020) and SDARTS (Chen & Hsieh, 2020) (see Table 2). These results again confirm that the eigenvalues are not necessarily a good indicator for finding better-performing models.

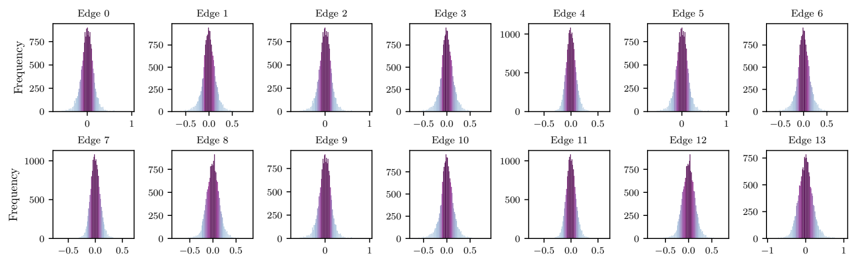


Figure 7: The Gaussian-like distribution of output features on all *skip* edges in the original DARTS.

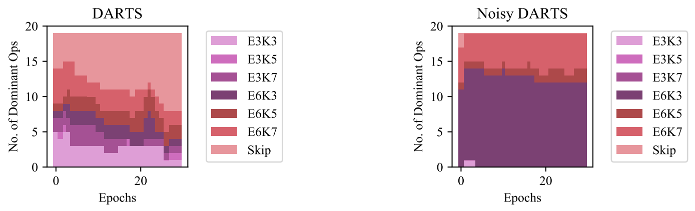


Figure 8: Stacked plot of dominant operations during searching on ImageNet. The inferred model of DARTS (left) obtains 66.4% accuracy on ImageNet, while NoisyDARTS (right) obtains 76.1%.

D NoisyDARTS architectures

D.1 Models searched on CIFAR-10 in the DARTS search space

We plot all the best models in different configurations of searching from Figure 11 to Figure 20.

D.2 Models searched on CIFAR-10 in the reduced search spaces of RDARTS

We plot them in Figure 21 and Figure 22.

D.3 GCN Models searched on ModelNet10

They are depicted in Figure 23.

Table 12: Best NoisyDARTS architecture genotypes searched on CIFAR-10 under biased noise

(μ, σ)	Architecture Genotype
(0.5, 0.1)	Genotype(normal=[('sep_conv_3x3', 0), ('sep_conv_3x3', 1), ('skip_connect', 0), ('dil_conv_3x3', 2), ('dil_conv_5x5', 3), ('dil_conv_3x3', 2), ('dil_conv_3x3', 4), ('dil_conv_3x3', 3)], normal_concat=range(2, 6), reduce=[('max_pool_3x3', 1), ('skip_connect', 0), ('skip_connect', 2), ('avg_pool_3x3', 1), ('skip_connect', 2), ('avg_pool_3x3', 1), ('dil_conv_5x5', 3), ('skip_connect', 4)], reduce_concat=range(2, 6))
(1.0, 0.1)	Genotype(normal=[('sep_conv_3x3', 0), ('sep_conv_3x3', 1), ('skip_connect', 1), ('skip_connect', 0), ('dil_conv_3x3', 3), ('dil_conv_5x5', 2), ('dil_conv_5x5', 4), ('dil_conv_3x3', 3)], normal_concat=range(2, 6), reduce=[('max_pool_3x3', 0), ('sep_conv_3x3', 1), ('max_pool_3x3', 0), ('dil_conv_5x5', 2), ('skip_connect', 3), ('skip_connect', 2), ('skip_connect', 3), ('skip_connect', 4)], reduce_concat=range(2, 6))
(0.5, 0.2)	Genotype(normal=[('sep_conv_3x3', 0), ('sep_conv_3x3', 1), ('skip_connect', 0), ('dil_conv_5x5', 3), ('dil_conv_5x5', 2), ('dil_conv_5x5', 4), ('dil_conv_3x3', 1)], normal_concat=range(2, 6), reduce=[('avg_pool_3x3', 0), ('skip_connect', 1), ('avg_pool_3x3', 0), ('skip_connect', 2), ('avg_pool_3x3', 0), ('dil_conv_3x3', 3), ('dil_conv_5x5', 4), ('avg_pool_3x3', 0)], reduce_concat=range(2, 6))
(1.0, 0.2)	Genotype(normal=[('sep_conv_3x3', 1), ('skip_connect', 0), ('dil_conv_5x5', 2), ('sep_conv_3x3', 1), ('dil_conv_3x3', 3), ('dil_conv_3x3', 2), ('dil_conv_3x3', 4), ('dil_conv_5x5', 3)], normal_concat=range(2, 6), reduce=[('max_pool_3x3', 0), ('max_pool_3x3', 1), ('max_pool_3x3', 0), ('max_pool_3x3', 1), ('dil_conv_5x5', 3), ('max_pool_3x3', 0), ('dil_conv_5x5', 3), ('avg_pool_3x3', 0)], reduce_concat=range(2, 6))

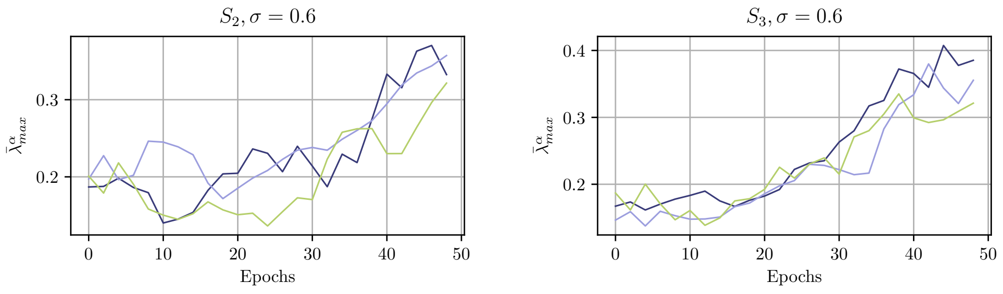


Figure 9: Evolution of maximal Hessian eigenvalue when searching with NoisyDARTS on two reduced search spaces S_2 and S_3 proposed by Zela et al. (2020). Compared with RDARTS, the eigenvalues still have a trend of increasing. Notice that better models can be found $3\times$ faster than RDARTS (they run four times to get the best model while we produce better ones at each single run).

Table 13: Test accuracy and the maximum Hessian eigenvalue (in the final searching epoch) of NoisyDARTS models searched with different σ in the reduced search spaces of RobustDARTS on CIFAR-10. Notice here we train models in S_2 with the same settings as in S_3. It's interesting to see that λ^{α}_{max} = 0.418 in S_3 is the best model with 97.47% top-1 accuracy. However, such similar value in Zela et al. (2020) indicates a failure (94.70%) under the same setting

Space	σ	Test acc. (%)			Params (M)			λ^{α}_{max}		
		seed 1	seed 2	seed 3	seed 1	seed 2	seed 3	seed 1	seed 2	seed 3
S_2	0.6	97.43	97.32	**97.46**	3.59	3.26	3.26	0.260	0.349	0.309
	0.8	97.30	97.39	97.37	3.62	3.62	3.62	0.133	0.270	0.429
	1.0	97.35	97.34	97.25	4.34	3.98	3.98	0.119	0.171	0.295
S_3	0.6	97.32	**97.47**	97.28	3.62	3.98	3.62	0.345	0.418	0.290
	0.8	97.32	97.24	97.27	3.98	3.62	3.26	0.393	0.327	0.336
	1.0	97.27	97.41	97.34	4.34	3.98	3.98	0.289	0.341	0.379

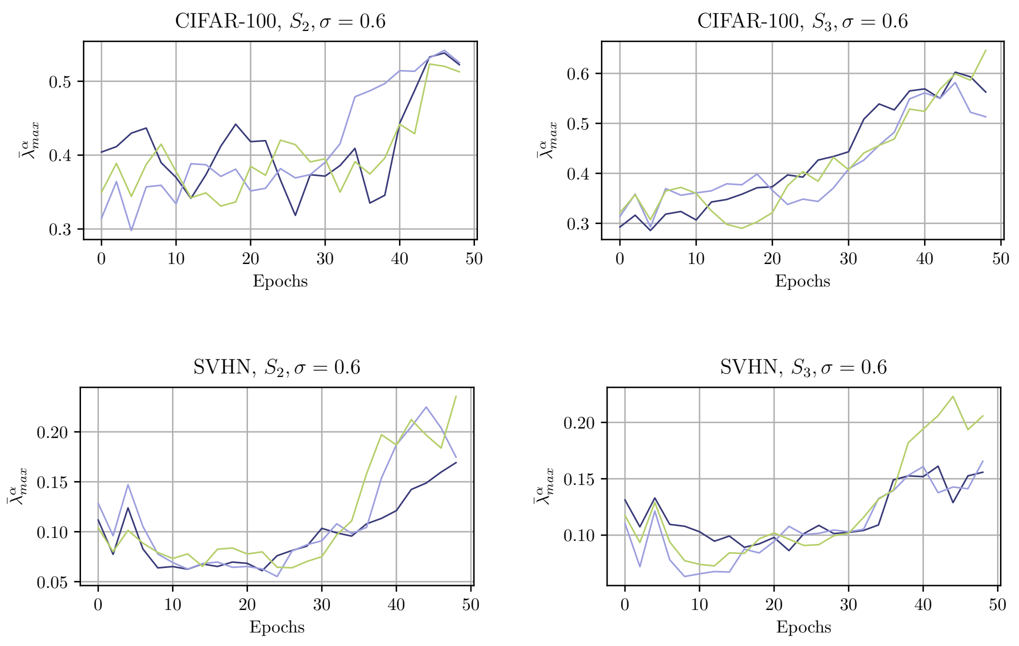
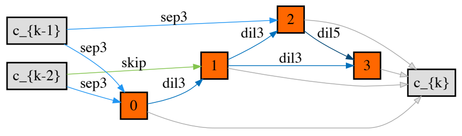
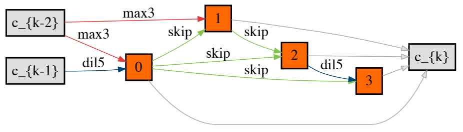


Figure 10: Evolution of maximal Hessian eigenvalue when searching with NoisyDARTS on CIFAR-100 and SVHN, in two reduced search spaces S_2 and S_3 proposed by Zela et al. (2020).

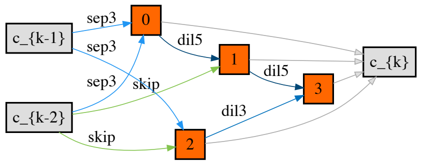


(a) Normal cell

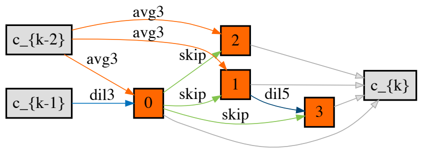


(b) Reduction cell

Figure 11: NoisyDARTS-a cells searched on CIFAR-10.

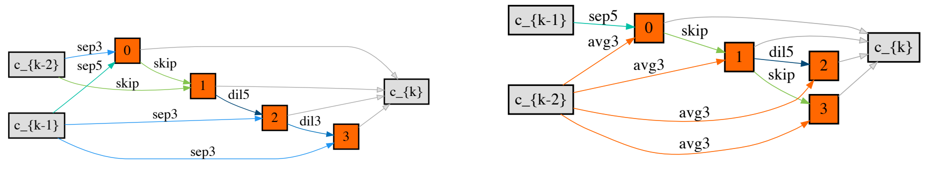


(a) Normal cell



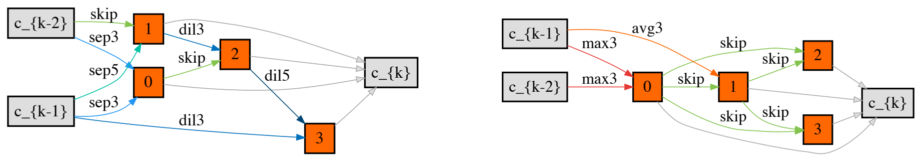
(b) Reduction cell

Figure 12: NoisyDARTS-b cells searched on CIFAR-10 with additive Gaussian noise, $\mu = 0$, $\sigma = 0.1$.



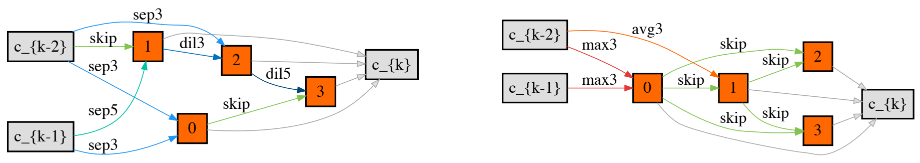
(a) Normal cell (b) Reduction cell

Figure 13: NoisyDARTS-c cells searched on CIFAR-10 with additive uniform noise, $\mu = 0$, $\sigma = 0.2$.



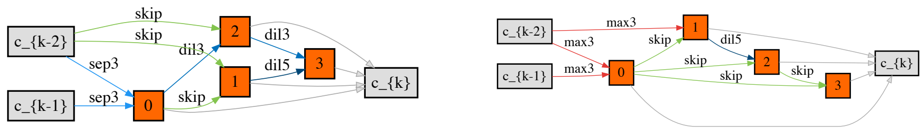
(a) Normal cell (b) Reduction cell

Figure 14: NoisyDARTS-d cells searched on CIFAR-10 with additive uniform noise, $\mu = 0$, $\sigma = 0.1$.



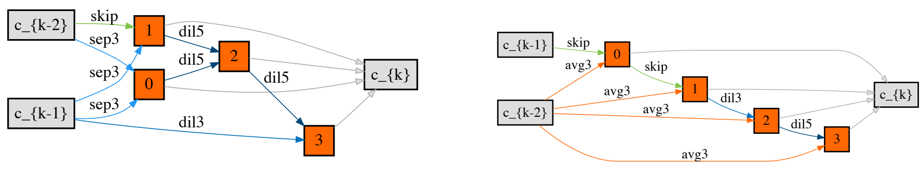
(a) Normal cell (b) Reduction cell

Figure 15: NoisyDARTS-e cells searched on CIFAR-10 with multiplicative Gaussian noise, $\mu = 0$, $\sigma = 0.2$.



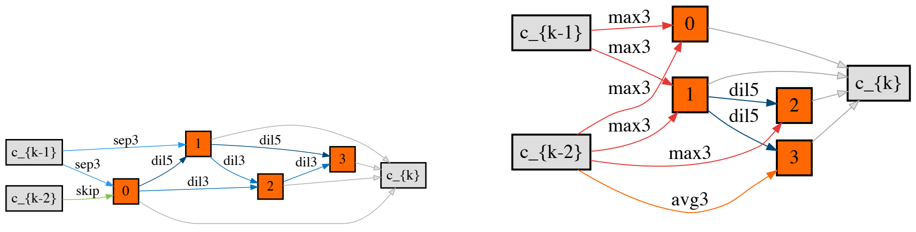
(a) Normal cell (b) Reduction cell

Figure 16: NoisyDARTS-f cells searched on CIFAR-10 with multiplicative Gaussian noise, $\mu = 0$, $\sigma = 0.1$.



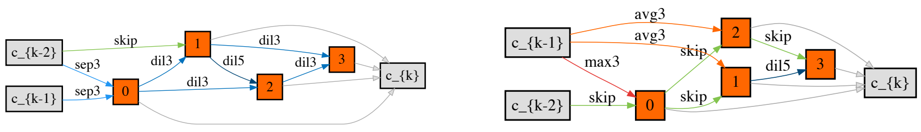
(a) Normal cell (b) Reduction cell

Figure 17: NoisyDARTS-g cells searched on CIFAR-10 with additive Gaussian noise, $\mu = 0.5$, $\sigma = 0.2$.



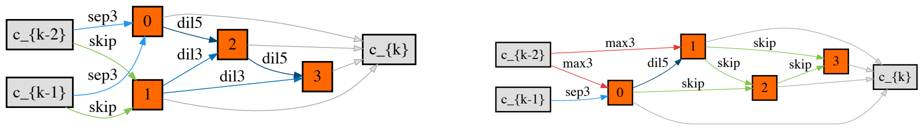
(a) Normal cell (b) Reduction cell

Figure 18: NoisyDARTS-h cells searched on CIFAR-10 with additive Gaussian noise, $\mu = 1.0$, $\sigma = 0.2$.



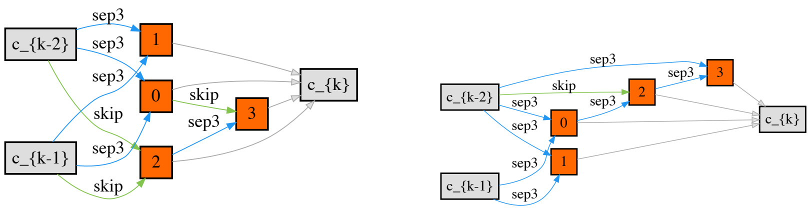
(a) Normal cell (b) Reduction cell

Figure 19: NoisyDARTS-i cells searched on CIFAR-10 with additive Gaussian noise, $\mu = 0.5$, $\sigma = 0.1$.



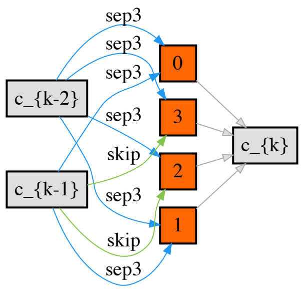
(a) Normal cell (b) Reduction cell

Figure 20: NoisyDARTS-j cells searched on CIFAR-10 with additive Gaussian noise, $\mu = 1.0$, $\sigma = 0.1$.

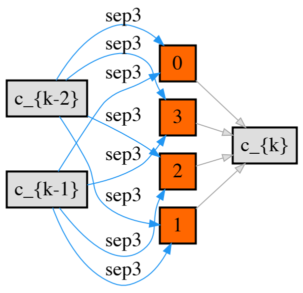


(a) Normal cell (b) Reduction cell

Figure 21: NoisyDARTS cells searched on CIFAR-10 with additive Gaussian noise $\mu = 0$, $\sigma = 0.6$, in S_2 of RobustDARTS.

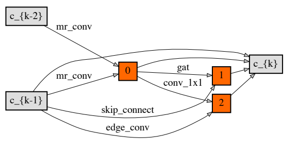


(a) Normal cell

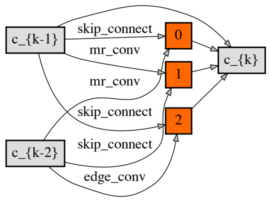


(b) Reduction cell

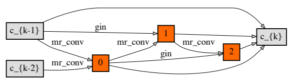
Figure 22: NoisyDARTS cells searched on CIFAR-10 with additive Gaussian noise $\mu = 0$, $\sigma = 0.6$, in S_3 of RobustDARTS.



(a) $\sigma = 0.2$



(b) $\sigma = 0.3$



(c) $\sigma = 0.4$

Figure 23: NoisyDARTS GCN cells searched on ModelNet-10 with additive Gaussian noise $\mu = 0$.