Supplementary of "Noisy Differentiable Architecture Search"

BMVC 2021 Submission # 1657

1 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 (main text), the expectation of the gradient under multiplicative noise can be written as:

$$\begin{aligned} \mathbb{E}\left[\nabla_{skip}\right] &= \mathbb{E}\left[\frac{\partial \mathcal{L}}{\partial y}\frac{\partial f(\alpha_s)}{\partial \alpha_s}\left(x'\right)\right] \\ &\approx \frac{\partial \mathcal{L}}{\partial y^{\star}}\frac{\partial f(\alpha_s)}{\partial \alpha_s}\left(x \cdot \mathbb{E}\left[\tilde{x}\right]\right). \end{aligned} \tag{1}$$

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

2 Algorithm

Algorithm 1 NoisyDARTS-OFS (default and recommended)

1: **Input:** Architecture parameters $\alpha_{i,j}$, network weights w , noise's standard variance σ, $Epoch_{max}$.
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 α by $\nabla_\alpha \mathcal{L}_{val}(w, \alpha)$
6: **end while**
7: Derive the final architecture according to learned α.

We give the NFA version of NoisyDARTS in Algorithm 2.

This work was done when both authors were with Xiaomi.
© 2021. The copyright of this document resides with its authors.
It may be distributed unchanged freely in print or electronic forms.

Algorithm 2 NoisyDARTS-NFA

1: **Input:** Architecture parameters $\alpha_{i,j}$, network weights w , noise's standard variance σ, $Epoch_{max}$.
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 α by $\nabla_\alpha \mathcal{L}_{val}(w, \alpha)$
6: **end while**
7: Derive the final architecture according to learned α.

3 More Experiments and Details

3.1 More Ablation Studies

Gaussian noise vs. uniform noise According to the analysis of Section 3.2 in the main text, 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 1. 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 Figure 1), a Gaussian noise $\tilde{x} \sim \mathcal{N}(0, \sigma_2^2)$ is preferred since the additive result shares the similar statistics, i.e., $x + \tilde{x} \sim N(\mu_1, \sigma_1^2 + \sigma_2^2)$.

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	97.35±0.23
Uniform	0.0	0.2	97.15±0.23

Table 1: Experiments on different types of noise. Each search is run 8 times

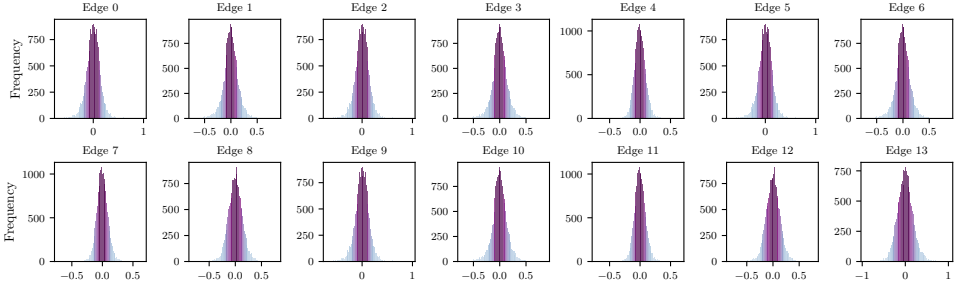


Figure 1: The Gaussian-like distribution of output features on all *skip* edges in the original DARTS.

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 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.

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	97.35±0.23
Multiplicative	1.0	0.2	97.22±0.23

Table 2: Experiments on different mixing operations. Each search is run 8 times

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 4 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 in the main text, 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.

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 a 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.

Searching GCN Architectures on ModelNet10.

3.2 Training Setting on Transferred Results on CIFAR-10

For transferred learning, we train the ImageNet-pretrained NoisyDARTS-A on CIFAR-10 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 [15].

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	**97.35±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

Table 3: Ablation on additive Gaussian noise on CIFAR-10 (each search is run 8 times)

Setting	CIFAR-10		CIFAR-100		ImageNet16-120	
	valid	test	valid	test	valid	test
DARTS w/ skip	39.77±0.00	54.30±0.00	15.03±0.00	15.61±0.00	16.43±0.00	16.32±0.00
DARTS w/o skip	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**

Table 4: Removing skip connection from search space on NAS-Bench-201.

Backbones	Params (M)	Acc (%)	AP (%)	AP_{50} (%)	AP_{75} (%)	AP_S (%)	AP_M (%)	AP_L (%)
MobileNetV2	3.4	72.0	28.3	46.7	29.3	14.8	30.7	38.1
SingPath NAS	4.3	75.0	30.7	49.8	32.2	15.4	33.9	41.6
MobileNetV3	5.4	75.2	29.9	49.3	30.8	14.9	33.3	41.1
MnasNet-A2	4.8	75.6	30.5	50.2	32.0	16.6	34.1	41.1
SCARLET-A	6.7	76.9	31.4	51.2	33.0	16.3	35.1	41.8
MixNet-M	5.0	77.0	31.3	51.7	32.4	17.0	35.0	41.9
FairNAS-A	5.9	77.5	32.4	52.4	33.9	17.2	36.3	43.2
NoisyDARTS-A	5.5	**77.9**	**33.1**	**53.4**	**34.8**	**18.5**	**36.6**	**44.4**

Table 5: COCO Object detection of various drop-in backbones.

3.3 Training Results on CIFAR-100

We show NoisyDARTS models searched in the DARTS search space and trained on CIFAR-100 in Table 8. We set the initial channel as 36 and the number of layers as 20.

3.4 Training Settings on COCO Object Detection

We use the MMDetection tool box since it provides a good implementation for various detection algorithms [1]. Following the same training setting as [10], all models in Table 5 are trained and evaluated on the COCO dataset for 12 epochs. The learning rate is initialized as 0.01 and decayed by $0.1\times$ at epoch 8 and 11.

3.5 Training 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.

3.6 Detailed Settings on NAS-Bench-201

For NAS-Bench-201 experiments, we adapt the code from [5]. We only use the first-order DARTS optimization. We track the running statistics for batch normalization to be the same as DARTS [12]. Each setting is run 3 times to obtain the average. We use a noise of $\sigma = 0.8$ regarding this particular search space.

3.7 Relations to Other Work

Comparison with PNI. PNI [7] uses parametric noise to boost adversarial training. In contrast, we inject the fixed noise at the output of candidate operations and smooth the

Method	CIFAR-10		CIFAR-100		ImageNet-16	
	val	test	val	test	val	test
DARTS-V1	39.77±0.00	54.30±0.00	15.03±0.00	15.61±0.00	16.43±0.00	16.32±0.00
ES, σ=0.2	39.77±0.00	54.30±0.00	15.03±0.00	15.61±0.00	16.43±0.00	16.32±0.00
ES, σ=0.4	49.27±16.46	59.84±9.59	22.87±13.59	23.39±13.48	17.24±1.40	17.01±1.20
NFA	88.17±2.02	91.60±1.74	67.71±2.35	68.26±1.59	41.91±2.00	41.57±2.59
OFS	**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 6: Comparison of NoisyDARTS (NFA, ES, OFS) on NAS-Bench-201.

Methods	Params (M)	OA (%)
SGAS (Cri. 1 avg.)	8.78	92.69±0.20
SGAS (Cri. 1 best)	8.63	92.87
NoisyDARTS ($\sigma = 0.3$ 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

Table 7: 3D classification on ModelNet40. OA: overall accuracy

loss 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).

3.8 Transferred Results

Transferred results on object detection. We further evaluate the transferability of our searched models on the COCO objection task [9]. Particularly, we utilize a drop-in replacement for the backbone based on Retina [10]. As shown in Table 5 (supplementary), our model obtains the best transferability than other models under the mobile settings. Detailed setting is provided in Section 3.4 (supplementary).

Transferring ImageNet models to CIFAR-10. We transferred our model NoisyDARTS-A 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 3.2 (Supplementary).

3.9 Evolution of NoisyDARTS architectural parameters

We plot the evolution of architectural parameters during the NoisyDARTS optimization in Figure 2. 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 [8] and [3], we don't set any hard limits for it. We also don't compute expensive Hessian eigenspectrum [16] as a type of regularization for skip connections. Neither do we use Scheduled DropPath [17] 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.

Models	Params (M)	Error (%)	Cost GPU Days
ResNet [6]	1.7	22.10$^{\diamond}$	-
AmoebaNet [14]	3.1	18.93$^{\diamond}$	3150
PNAS [11]	3.2	19.53$^{\diamond}$	150
ENAS [13]	4.6	19.43$^{\diamond}$	0.45
DARTS [7]	-	20.58±0.44*	0.4
GDAS [4]	3.4	18.38	0.2
P-DARTS [3]	3.6	17.49‡	0.3
R-DARTS [16]	-	18.01±0.26	1.6
NoisyDARTS	4.7	**16.28**	0.4

Table 8: Comparison of searched models on CIFAR-100. $^{\diamond}$: Reported by [4], *: Reported by [16], ‡:Rerun their code.

3.10 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 eigenvalue trajectories of our method in the reduced search space, which are shown in Figure 5.

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 [16]. Compared with their best effort L2 regularization [16] and another method SDARTS [2] based on implicit regularization of Hessian norm, we also have better performance in S_2 and comparable performance in S_3 (see Table 2 in the main text). Notice we only use **3x fewer** searching cost. This reassures our observation that the Hessian norm [16] may not be an ideal indicator of performance collapse, because it rejects good models by mistake, as illustrated in Figure 6.

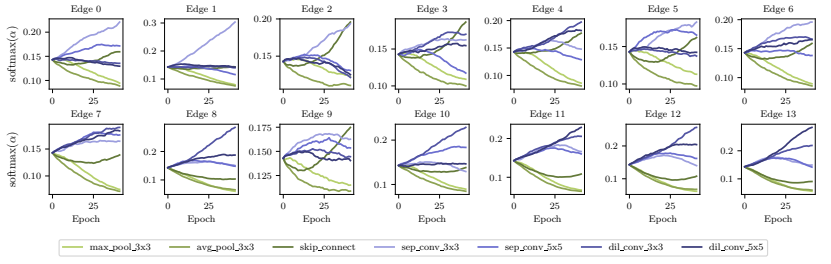
3.11 More Details on Reduced RobustDARTS Experiments

Like on CIFAR-10, we repeatedly find the Hessian eigenvalues are both growing when searching with NoisyDARTS on CIFAR-100 and SVHN datasets (see Figure 7), but models derived from these searching runs still outperform or are comparable to those from regularized methods like RDARTS [16] and SDARTS [2] (see Table 2 in the main text). These results again confirm that the eigenvalues are not necessarily a good indicator for finding better-performing models.

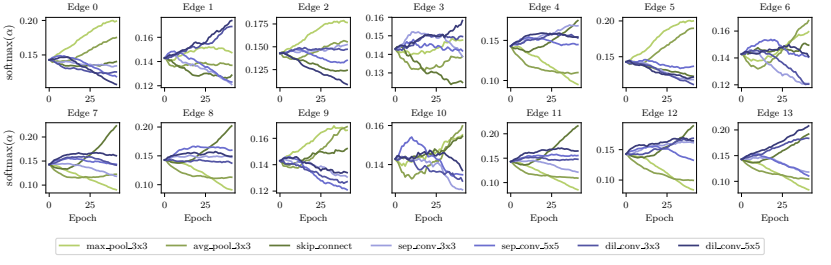
4 NoisyDARTS architectures

4.1 Models searched on CIFAR-10 in the DARTS search space

We plot all the best models in different configurations of searching from Figure 8 to Figure 17.



(a) Normal cell



(b) Reduction cell

Figure 2: Evolution of architectural weights during the NoisyDARTS searching phase on CIFAR-10. Skip connections in normal cells are largely suppressed.

Figure 3: NoisyDARTS-A searched on ImageNet. Colors represent different stages.

4.2 Models searched on CIFAR-10 in the reduced search spaces of RDARTS

We plot them in Figure 18 and Figure 19.

4.3 GCN Models searched on ModelNet10

They are depicted in Figure 20.

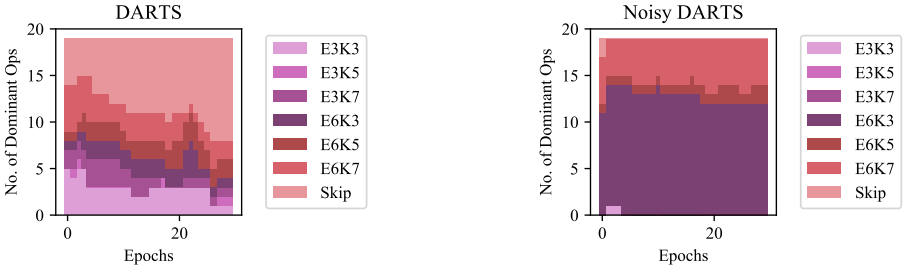


Figure 4: Stacked plot of dominant operations during searching on ImageNet. The inferred model of DARTS (left) obtains 66.4% accuracy on ImageNet, while NoisyDARTS (right) obtains 76.1%.

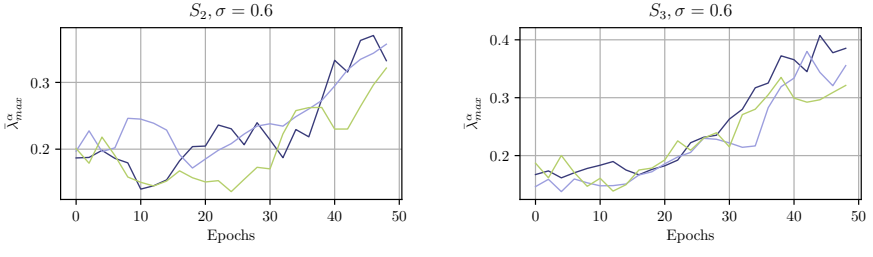


Figure 5: Evolution of maximal Hessian eigenvalue when searching with NoisyDARTS on two reduced search spaces S_2 and S_3 proposed by [16]. Compared with RDARTS, the eigenvalues still have a trend of increasing. Notice that better models can be found $3\times$ faster than RDARTS (they run four times to get the best model while we produce better ones at each single run).

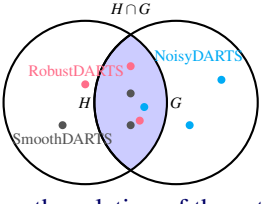


Figure 6: Exemplary illustration on the relation of the set of models. Set H means models found with low Hessian norms. Set G are the models with better test accuracy. RobustDARTS's Hessian norm criterion [16] tends to reject a part of good models, e.g. blue models found by NoisyDARTS that are not in $H \cap G$.

Space	σ	Test acc. (%)			Params (M)			λ^{α}_{max}		
		seed			seed			seed		
		1	2	3	1	2	3	1	2	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

Table 9: 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 $\lambda^{\alpha}_{max} = 0.418$ in S_3 is the best model with 97.47% top-1 accuracy. However, such similar value in [16] indicates a failure (94.70%) under the same setting

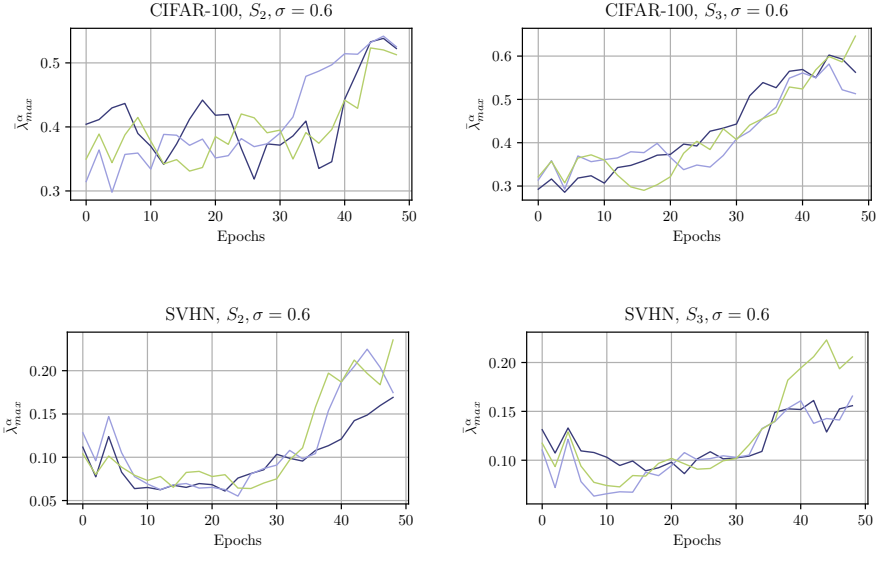
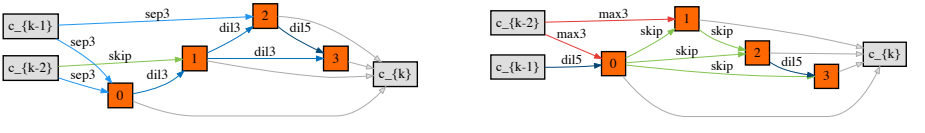
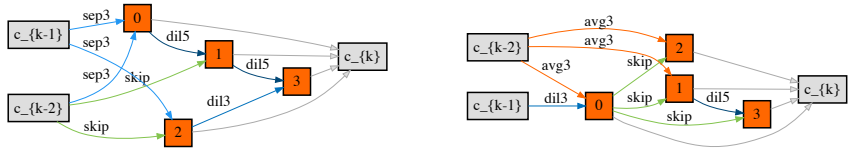


Figure 7: Evolution of maximal Hessian eigenvalue when searching with NoisyDARTS on CIFAR-100 and SVHN, in two reduced search spaces S_2 and S_3 proposed by [16].



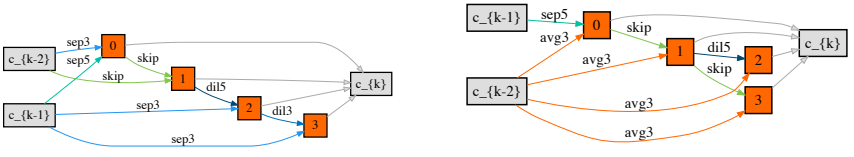
(a) Normal cell (b) Reduction cell

Figure 8: NoisyDARTS-a cells searched on CIFAR-10.



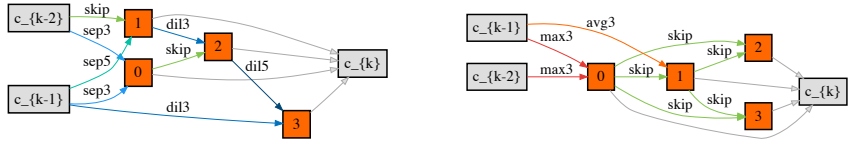
(a) Normal cell (b) Reduction cell

Figure 9: NoisyDARTS-b cells searched on CIFAR-10 with additive Gaussian noise, $\mu = 0$, $\sigma = 0.1$.



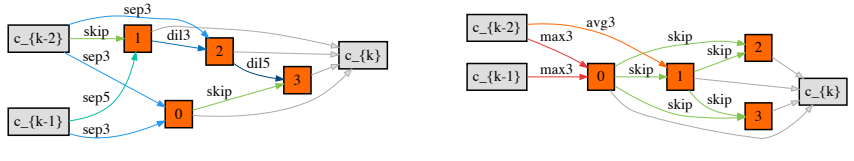
(a) Normal cell (b) Reduction cell

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



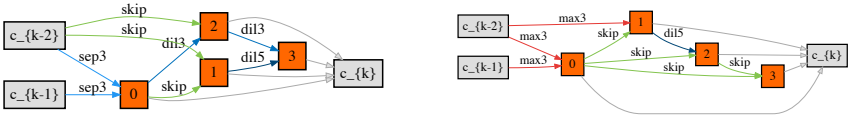
(a) Normal cell (b) Reduction cell

Figure 11: NoisyDARTS-d cells searched on CIFAR-10 with additive uniform noise, $\mu = 0$, $\sigma = 0.1$.



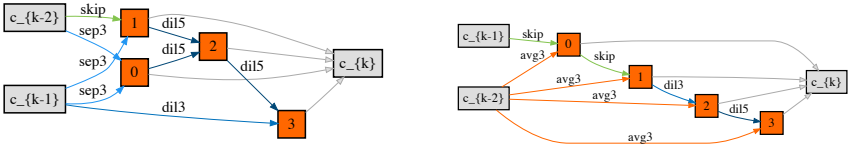
(a) Normal cell (b) Reduction cell

Figure 12: NoisyDARTS-e cells searched on CIFAR-10 with multiplicative Gaussian noise, $\mu = 0$, $\sigma = 0.2$.



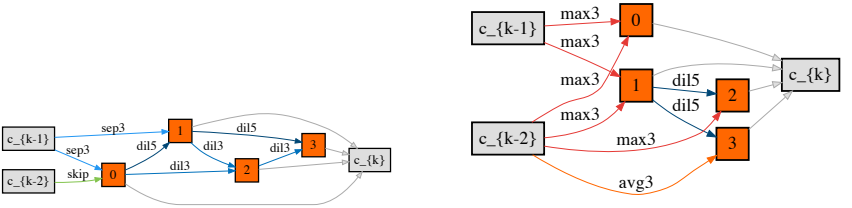
(a) Normal cell (b) Reduction cell

Figure 13: NoisyDARTS-f cells searched on CIFAR-10 with multiplicative Gaussian noise, $\mu = 0$, $\sigma = 0.1$.



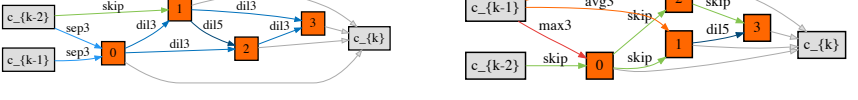
(a) Normal cell (b) Reduction cell

Figure 14: NoisyDARTS-g cells searched on CIFAR-10 with additive Gaussian noise, $\mu = 0.5$, $\sigma = 0.2$.



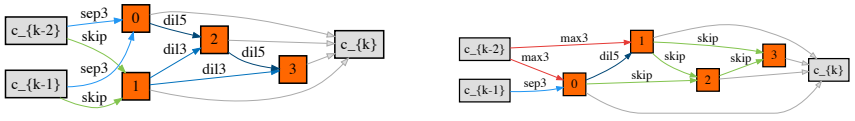
(a) Normal cell (b) Reduction cell

Figure 15: NoisyDARTS-h cells searched on CIFAR-10 with additive Gaussian noise, $\mu = 1.0$, $\sigma = 0.2$.



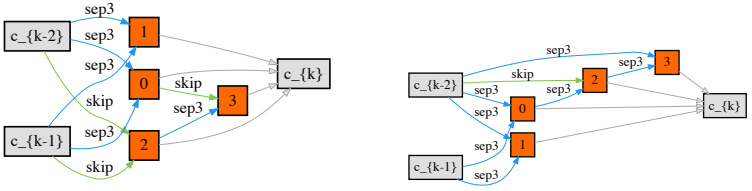
(a) Normal cell (b) Reduction cell

Figure 16: NoisyDARTS-i cells searched on CIFAR-10 with additive Gaussian noise, $\mu = 0.5$, $\sigma = 0.1$.



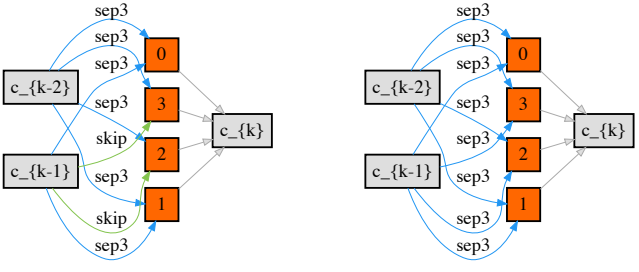
(a) Normal cell (b) Reduction cell

Figure 17: NoisyDARTS-j cells searched on CIFAR-10 with additive Gaussian noise, $\mu = 1.0$, $\sigma = 0.1$.



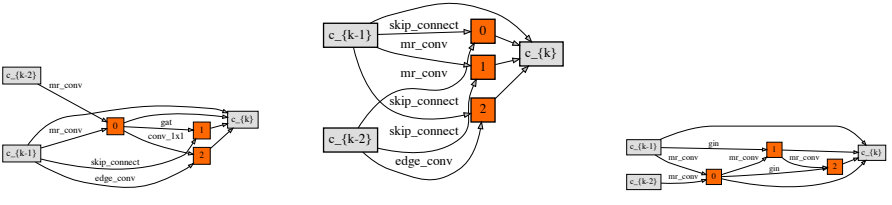
(a) Normal cell (b) Reduction cell

Figure 18: 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 19: 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 20: NoisyDARTS GCN cells searched on ModelNet-10 with additive Gaussian noise $\mu = 0$.

References

[1] 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.

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

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

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

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

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

[7] 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.

[8] 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.

[9] 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.

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

[11] 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.

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

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

[14] 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.

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

[16] 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.

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