Neighborhood-Aware Neural Architecture Search (Supplementary Materials)

Xiaofang Wang, Shengcao Cao*,
Mengtian Li*, Kris M. Kitani
{xiaofan2,shengcao,mtli,kkitani}@cs.cmu.edu

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA, USA

A Aggregation Function

A.1 More Choices for Aggregation Function

Our formulation aims to identify flat minima in the search space based on the aggregated performance $g(f(\mathcal{N}(\alpha)))$ over the neighborhood. The aggregation function $g(\cdot)$ needs to be properly set such that minimzing $g(f(\mathcal{N}(\alpha)))$ results in an architecture α that is a local minimum and at the same time has a flat neighborhood.

The flatness of the neighborhood of α is determined by how much the performance (*e.g.*, validation loss) of its neighboring architectures varies compared to α itself. Intuitively, when α is a flat minimum, its neighboring architectures should perform similarly to α. However, when α is a sharp minimum, the loss of architectures around α increases drastically compared to α. Although the formal definition of flatness or sharpness of a local minimum is not exactly the same in previous work [1, 5, 8, 9, 17], they all share this intuition.

We discuss possible choices for the aggregation function $g(\cdot)$:

- mean, median or max.

  The architectures around a sharp minimum tend to high much higher loss compared to this minimum. Therefore, the mean validation loss of architectures around a flat minimum is expected to be lower than those around a sharp minimum. Minimizing mean$(f(\mathcal{N}(\alpha)))$ encourages the convergence to an architecture α whose neighbors in $\mathcal{N}(\alpha)$ all have a low loss, which implies that α is a flat minima. This makes mean a valid choice. For a similar reason, median and max are also valid choices.

  Setting $g(\cdot)$ as mean or max also aligns well with previous work on flat minima. [1] propose an objective function for training neural networks so that flat minima are preferred during optimization. Their objective can be interpreted as a weighted average of the (transformed) function values of data points around the local minima, which inspires us to consider mean as one of the choices for $g(\cdot)$. [9] use the largest function value that can be attained in the neighborhood of a local minimum to characterize how sharp the minimum is, which leads us to set $g(\cdot)$ as max.

- Variance.

* indicates equal contribution.

For an architecture α, we can measure its flatness with the variance (standard deviation) of the performance of its neighbors in $\mathcal{N}(\alpha)$. Let $\sigma(f(\mathcal{N}(\alpha)))$ denote the standard deviation of the performance (*e.g.*, validation loss) of architectures in $\mathcal{N}(\alpha)$. But simply minimizing $\sigma(f(\mathcal{N}(\alpha)))$ can only result in an α with a flat neighborhood, but cannot guarantee that α is a local minimum (*e.g.*, have a low validation loss). So we propose the following variance-based aggregation function $g(f(\mathcal{N}(\alpha))) = f(\alpha) + \lambda\sigma(f(\mathcal{N}(\alpha)))$ that takes both the performance of α and the flatness of its neighborhood into account, where λ is a hyper-parameter to balance the performance $f(\alpha)$ and the flatness $\sigma(f(\mathcal{N}(\alpha)))$.

A.2 Aggregation Function in Differentiable Architecture Search

When applying our formulation to differentiable NAS methods, $g(\cdot)$ needs to be differentiable, which immediately rules out median. Our default choice is mean and we provide an outline of NA-DARTS using mean in the main text.

Both mean and the variance-based aggregation function are differentiable. We prefer mean because it requires fewer GPU memory. Theoretically, when computing $\nabla_\alpha g(f(\mathcal{N}(\alpha)))$, we need to keep all architectures in $\mathcal{N}(\alpha)$ in GPU. But when $g(\cdot) = \text{mean}$, we can compute $\nabla_\alpha f(\alpha')$ separately for each neighbor $\alpha' \in \mathcal{N}(\alpha)$. Since PyTorch [15] automatically accumulates the gradient in multiple backward passes, computing $\nabla_\alpha f(\alpha')$ separately is equivalent as computing $\nabla_\alpha \text{mean}(f(\mathcal{N}(\alpha)))$. Therefore, when using mean, we only need to keep one architecture in GPU. This requires much fewer GPU memory than the variance-based aggregation function.

We prefer mean over max due to its superior empirical performance. When using max, Eq. 5 becomes a minimax optimization problem and one can approximate the gradient of the objective using Danskin's Theorem [3]. Same as mean, max also only needs to keep one architecture in GPU (see following text for more details).

Algorithm 1 Neighborhood-Aware DARTS

Input: Number of steps T. Number of neighbors n_{nbr}. Initial architecture α and weights w.
for $t = 1, 2, \ldots, T$ **do**
  Sample a batch of training data X_{train} and a batch of validation data X_{val}.
  Sample n_{nbr} neighboring architectures of α: $\mathcal{N}(\alpha)$.
  if $g(\cdot) == \max$ **then**
    Compute $\bar{\alpha} = \arg\max_{\alpha' \in \mathcal{N}(\alpha)} \mathcal{L}_{\text{val}}(w, \alpha')$ on X_{val}.
    Compute $\nabla_\alpha \mathcal{L}_{\text{val}}(w, \bar{\alpha})$ on X_{val}; update α by descending $\nabla_\alpha \mathcal{L}_{\text{val}}(w, \bar{\alpha})$.
  else if $g(\cdot) == \text{mean}$ **then**
    Compute $\nabla_\alpha \frac{\sum_{\alpha' \in \mathcal{N}(\alpha)} \mathcal{L}_{\text{val}}(w, \alpha')}{|\mathcal{N}(\alpha)|}$ on X_{val}; update α by descending $\nabla_\alpha \frac{\sum_{\alpha' \in \mathcal{N}(\alpha)} \mathcal{L}_{\text{val}}(w, \alpha')}{|\mathcal{N}(\alpha)|}$.
  end if
  Compute $\nabla_w \mathcal{L}_{\text{train}}(w, \alpha)$ on X_{train}; update w by descending $\nabla_w \mathcal{L}_{\text{train}}(w, \alpha)$.
end for
Derive the final architecture based on the learned α.

Algorithm 2 Neighborhood-Aware Random Search

Input: Number of steps T. Number of neighbors n_{nbr}.
for $t = 1, 2, \ldots, T$ **do**
  Randomly sample an architecture from $\mathcal{A}$: α.
  Sample n_{nbr} neighboring architectures of α: $\mathcal{N}(\alpha)$.
  Train the n_{nbr} architectures and compute $g(f(\mathcal{N}(\alpha)))$.
  Let $\alpha^* = \alpha$ if $g(f(\mathcal{N}(\alpha))) < g(f(\mathcal{N}(\alpha^*)))$.
end for
Return the optimal architecture α^*.

A.2.1 Using max in NA-DARTS

For completeness, we describe details of using max in NA-DARTS. After setting $g(\cdot)$ as max, Eq. 5 becomes a minimax optimization. According to Danskin's Theorem [3], we can approximate the gradient $\nabla_\alpha \max_{\alpha' \in \mathcal{N}(\alpha)} \mathcal{L}_{\text{val}}(w^*(\alpha'), \alpha')$ with $\nabla_\alpha \mathcal{L}_{\text{val}}(w^*(\bar{\alpha}), \bar{\alpha})$, where $\bar{\alpha}$ is the maximizer of the inner maximization problem $\max_{\alpha' \in \mathcal{N}(\alpha)} \mathcal{L}_{\text{val}}(w^*(\alpha'), \alpha')$. In practice, $w^*(\alpha')$ is approximated by the current network weights w. To compute the maximizer $\bar{\alpha}$, we simply compute the validation loss of each sampled neighboring architecture and choose the maximum one. We provide an outline of NA-DARTS in Algorithm 1, where we include steps for both cases ($g(\cdot) = \max$ or $g(\cdot) = \text{mean}$). As can seen from Algorithm 1, when using max, we only need to keep one architecture ($\bar{\alpha}$) in GPU during the gradient computation.

Solving the inner maximization problem $\max_{\alpha' \in \mathcal{N}(\alpha)} \mathcal{L}_{\text{val}}(w^*(\alpha'), \alpha')$ is the process of finding the worst-performing neighbor of α in its neighborhood. Sampling neighbors with the additive representation of neighbors (Eq. 6) might not always result in a neighbor α' that performs worse than α. So, we develop the following *multiplicative representation* of neighboring architectures. The multiplicative representation allows us to sample α' by changing a subset of operations in α to the zero operation or skip connection such that α' has a higher probability to perform worse than α. Let edge (i, j) be an edge to be perturbed and $r^{(i,j)}$ be a m-dim one-hot vector with $r_l^{(i,j)} = 1$ and $r_k^{(i,j)} = 0 (1 \leq k \leq m, k \neq l)$. We restrict l to be either the index of the zero operation or skip connection. With the one-hot vector $r^{(i,j)}$, $\alpha'^{(i,j)}$ is computed as:

$$\alpha_k'^{(i,j)} = \frac{r_k^{(i,j)} \alpha_k^{(i,j)}}{\sum r_k^{(i,j)} \alpha_k^{(i,j)}}. \tag{A}$$

Under the multiplicative representation, $\alpha'^{(i,j)}$ has the same value as $r^{(i,j)}$, which indicates that the edge (i, j) after perturbation chooses either the zero operation or skip connection. We empirically observe that max works better with the multiplicative representation than additive representation.

B Assumption Justification

B.1 Experimental Setup

We describe the detailed setup of our assumption justification experiments in Sec. 3.3. NAS-Bench-201 [6] provides a simulated environment for NAS experiments by conducting a thorough evaluation of all the candidate architectures (cells) in a pre-defined cell search space

(a) $f(\alpha)$ = CIFAR-10-Validation error after the 30^{th} epoch.

	CIFAR-10-Validation	CIFAR-10	CIFAR-100	ImageNet-16-120
Flat minima	18.39	**6.33**	**29.15**	**55.52**
Sharp minima	18.45	6.67	30.10	56.18

(b) $f(\alpha)$ = CIFAR-10-Validation error after the 60^{th} epoch.

	CIFAR-10-Validation	CIFAR-10	CIFAR-100	ImageNet-16-120
Flat minima	16.15	**6.28**	**29.15**	**55.51**
Sharp minima	16.43	6.91	30.56	57.31

(c) $f(\alpha)$ = CIFAR-10-Validation error after the 90^{th} epoch.

	CIFAR-10-Validation	CIFAR-10	CIFAR-100	ImageNet-16-120
Flat minima	14.55	**6.23**	**28.90**	**55.17**
Sharp minima	14.57	6.66	30.00	56.41

(d) $f(\alpha)$ = CIFAR-10-Validation error after the 120^{th} epoch.

	CIFAR-10-Validation	CIFAR-10	CIFAR-100	ImageNet-16-120
Flat minima	12.67	**6.13**	**28.59**	**55.11**
Sharp minima	12.81	6.33	29.28	55.53

Table A: Average error of flat-minima architectures and sharp-minima architectures. "CIFAR-10-Validation" refers to the average validation error on CIFAR-10 used in search. CIFAR-10, CIFAR-100 and ImageNet-16-120 refer to the average test error on each dataset. Flat minima and sharp minima obtain a similar validation error on CIFAR-10. However, flat minima consistently achieves lower test error than sharp minima on all three datasets.

on three datasets: CIFAR-10 [10], CIFAR-100 [10], and ImageNet-16-120 [6]. It contains the validation error (accuracy) of all the candidate architectures on CIFAR-10 after every training epoch, and the final test error on CIFAR-10, CIFAR-100, and ImageNet-16-120. ImageNet-16-120 is a subset and downsampled version of ImageNet [16] and contains about 158K images divided into 120 classes.

In our experiments, we set the distance threshold d to 1, so each architecture in the NAS-Bench-201 search space has 25 neighbors including itself. We search on CIFAR-10 and evaluate the found architectures on all three datasets, *i.e.*, $f(\alpha)$ is defined as the validation error on CIFAR-10. It is common in NAS to use early stopping or budgeted training during search [7, 12]. So, we use the CIFAR-10-Validation error after the 90^{th} epoch in the experiments, unless otherwise stated. Results for other epochs (*e.g.*, 30^{th}, 60^{th}, 120^{th}) lead to the same conclusion.

B.2 Flat Minima Generalize Better

We provide results for other epochs to show that flat minima in the architecture space generalize better than sharp minima. Specifically, we conduct the same experiments as Sec. 3.3.1 (Table 1a in the main text) with the CIFAR-10-Validation error after the 30^{th}, 60^{th} or 120^{th} epoch. As shown in Table A, results for all epochs (30^{th}, 60^{th}, 90^{th}, 120^{th}) demonstrate the

same pattern: the average validation error on CIFAR-10 of flat minima and sharp minima are similar; however, the average test error of flat minima is consistently lower than sharp minima on all three datasets, especially on CIFAR-100 and ImageNet-16-120.

B.3 Aggregated Performance Gives a Better Ranking of Architectures

We show that our criterion $g(f(\mathcal{N}(\alpha)))$ ranks architectures more accurately than the standard criterion $f(\alpha)$. To do that, we randomly sample 100 architectures from NAS-Bench-201 and rank these architectures according to our criterion $g(f(\mathcal{N}(\alpha)))$ or the standard criterion $f(\alpha)$, where $f(\cdot)$ is the validation error on CIFAR-10. Following [13], we evaluate the estimated ranking with the Kendall's Tau metric (the higher the better), which measures the correlation between the estimated ranking and ground truth ranking of architectures. The ground truth is obtained by sorting these architectures based on their test error. As the ground truth is specific to each dataset, we evaluate the estimated ranking on the three datasets separately.

We repeat the experiments for 10 times and report the mean and standard deviation of the Kendall's Tau value. Table 1b (main text) shows the ranking estimation results when $g(\cdot) = \text{mean}$. We provide the results for all the aggregation functions in Table B. For the variance-based aggregation function, we set λ to 1.0. All aggregation functions except max result in an more accurate ranking estimation of architectures than the standard criterion $f(\alpha)$.

	CIFAR-10	CIFAR-100	ImageNet-16-120
Baseline	0.66 ± 0.03	0.66 ± 0.02	0.64 ± 0.03
Ours - mean	$\mathbf{0.76 \pm 0.03}$	$\mathbf{0.77 \pm 0.03}$	$\mathbf{0.74 \pm 0.03}$
Ours - median	0.72 ± 0.03	0.72 ± 0.03	0.69 ± 0.03
Ours - max	0.53 ± 0.05	0.54 ± 0.05	0.56 ± 0.05
Ours - Variance	0.72 ± 0.02	0.73 ± 0.03	0.71 ± 0.02

Table B: Kendall's Tau (rank correlation) obtained by the standard criterion $f(\alpha)$ (baseline) and our criterion $g(f(\mathcal{N}(\alpha)))$ with different choices of $g(\cdot)$.

	Neighbor-Var	CIFAR-10	CIFAR-100	ImageNet-16-120
Baseline	5.58	6.45	29.45	55.79
Ours - mean	2.71	**6.09**	**28.32**	**54.75**
Ours - median	4.05	6.21	28.74	55.08
Ours - max	1.83	6.66	29.82	56.31
Ours - Variance	2.47	6.35	29.06	55.52

Table C: Neighborhood variance and test error of architectures found by by the standard criterion $f(\alpha)$ (baseline) and our criterion $g(f(\mathcal{N}(\alpha)))$ with different choices of $g(\cdot)$. Architectures found by the mean validation error ('Ours - mean') have a much smaller neighborhood variance than those found by the baseline criterion, and also achieve lower classification error on all three datasets.

B.4 Aggregated Performance Finds Flat Minima

We conduct quantitative analysis to show that optimizing the proposed criterion, *i.e.*, the aggregated performance over the neighborhood $g(f(\mathcal{N}(\alpha)))$, successfully finds flat minima. We select 100 architectures from NAS-Bench-201 with the lowest validation error (standard criterion) and another 100 architectures with the lowest aggregated validation error (proposed criterion) on CIFAR-10.

We measure the flatness of an architecture with its neighborhood variance: the variance of the search-time validation error of its neighboring architectures on CIFAR-10. A smaller variance indicates a flatter neighborhood. We summarize the neighborhood variance and test error of the found architectures in Table C. We observe that optimizing the mean validation error ('Ours - mean') can successfully help us find flat minima, as the found architectures have a much smaller neighborhood variance than those found by the baseline criterion, and also achieve lower classification error on all three datasets.

We also notice that when $g(\cdot) = \max$, the found architectures are not flat minima. Although these architectures have a flat neighborhood (low neighborhood variance), their classification performance is worse than architectures found by the baseline criterion. We think this is because when using max, the objective $g(f(\mathcal{N}(\alpha)))$ only considers the flatness of the neighborhood, but fails to characterize how well the architecture α performs.

C NA-RS

Experimental setup. An outline of NA-RS is provided in Algorithm 2. Same as the setup in the assumption justification experiments, we search on CIFAR-10 and evaluate on CIFAR-10 [10], CIFAR-100 [10], and ImageNet-16-120 [6]. The number of search steps T in NA-RS is set to 100. For fair comparison, the standard random search (baseline; denoted as 'RS') is run for $T \cdot n_{\text{nbr}}$ steps, so that RS and NA-RS train and evaluate the same number of architectures. We set the distance threshold d to 1, so the neighborhood contains 25 architectures including the reference architecture itself. We set n_{nbr} to 10 unless otherwise stated.

Ablation study. We provide an ablation study of the aggregation function in NA-RS in Table D and an ablation study of n_{nbr} in Table E. We see from Table D that mean and median achieve the best performance among all the choices for $g(\cdot)$. max performs the worst, which is consistent with the conclusion in Table B. As shown in Table E, performance obtained by $n_{\text{nbr}} = 10$ is close to $n_{\text{nbr}} = 25$, which indicates that sampling a subset of neighbors is a good approximation for the entire neighborhood.

	CIFAR-10	CIFAR-100	ImageNet-16-120
NA-RS - mean	6.39 ± 0.71	28.68 ± 1.75	55.02 ± 1.71
NA-RS - median	6.20 ± 0.35	28.33 ± 1.22	54.72 ± 0.96
NA-RS - max	6.73 ± 0.71	29.70 ± 1.61	56.96 ± 2.09
NA-RS - Variance	6.65 ± 0.97	29.06 ± 1.97	55.48 ± 2.41

Table D: Ablation study on the aggregation function in NA-RS. mean and median yield the lower test error among all the choices for $g(\cdot)$.

		CIFAR-10	CIFAR-100	ImageNet-16-120
NA-RS - mean	$n_{\text{nbr}} = 10$	6.39 ± 0.71	28.68 ± 1.75	55.02 ± 1.71
	$n_{\text{nbr}} = 25$	6.24 ± 0.39	28.24 ± 1.25	54.74 ± 1.73
NA-RS - median	$n_{\text{nbr}} = 10$	6.20 ± 0.35	28.33 ± 1.22	54.72 ± 0.96
	$n_{\text{nbr}} = 25$	6.18 ± 0.38	28.20 ± 1.27	54.40 ± 0.98

Table E: Ablation study on n_{nbr} in NA-RS. Sampling a subset of neighbors ($n_{\text{nbr}} = 10$) is a good approximation for the entire neighborhood ($n_{\text{nbr}} = 25$).

D NA-DARTS

D.1 Experimental Setup

Following DARTS [14], we search on CIFAR-10 [10] and evaluate on CIFAR-10 [10], CIFAR-100 [10] and ImageNet [16]. We use exactly the same setup as DARTS [14], including the cell search space, hyper-parameters, such as the learning rate and weight decay factor, and other experimental details. We split the training images in CIFAR-10 into two subsets of equal size, which are used as the training and validation images during search. We construct a network of 8 cells with an initial channel number as 16 and train the network for 50 epochs to learn α.

After the search is done, we derive the final architecture from the learned α using exactly the same procedure as DARTS. When evaluating the found architecture on CIFAR-10 and CIFAR-100, we build a network of 20 cells and train it for 600 epochs with batch size 96 and cutout [4]. For our NA-DARTS, We set the initial number of channels of the network such that it has a similar network size with DRATS and contains around 3M parameters.

When evaluating on ImageNet, we build a network of 14 cells. Following DARTS, the network is trained for 250 epochs with batch size 128. We set the initial number of channels such that the number of multiply-add operations in the network is fewer than 600M when the input is 224×224. Some NAS methods use a different training setup to train the found architecture on ImageNet. For example, DARTS+ [13] trains for 800 epochs and P-DARTS [2] uses a large batch size 1024 (need 8 V100 GPUs, infeasible to us). For fair comparison, we retrain the found architecture reported by the authors in their paper using the same training setup as DARTS.

For our NA-DARTS, we sample a subset of 10 neighbors in each step, *i.e.*, $n_{\text{nbr}} = 10$. The distance threshold d for neighborhood can be interpreted as the number of edges to be perturbed. As each cell in the DARTS search space has 14 edges, we set d to 6. The noise threshold ε in the additive representation is set to 0.1. All experiments are performed on a NVIDIA GeForce RTX 2080 Ti GPU.

D.2 Ablation Study

Aggregation function. We report the performance of NA-DARTS when using mean or max as the aggregation function in Table Fa. We observe that mean outperforms max, which is consistent with the conclusion in Table B. We also notice that mean consumes a longer search time than max. This is because when using mean, we need to back-propagate through every sampled neighboring architecture α', while we only need to back-propagate through one neighboring architecture $\bar{\alpha}$ when using max.

Distance threshold. We study the impact of the distance threshold of d in Table Fb, where we observe $d = 6$ achieves the best performance and $d = 4$ performs similarly with $d = 6$. Recall that the distance threshold d can be interpreted as the number of edges to be perturbed and the cell in the DARTS search space has 14 edges. We empirically find that when d becomes larger that 6, the neighborhood becomes too large and the performance drops.

(a) Impact of aggregation function.

	Test Error (%)		Param	Search Cost
	CIFAR-10	CIFAR-100	(M)	(GPU days)
max	2.80 ± 0.10	16.89 ± 0.31	3.1	0.5
mean	2.63 ± 0.12	16.48 ± 0.13	3.2	1.1

(b) Impact of d.

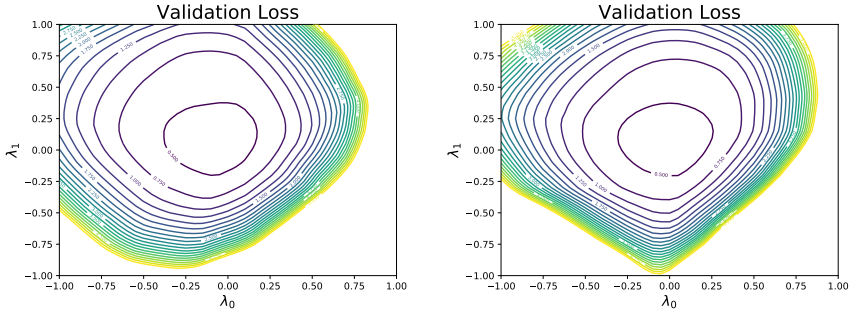
	Test Error (%)		Param
	CIFAR-10	CIFAR-100	(M)
$d = 2$	2.62 ± 0.08	16.90 ± 0.45	3.2
$d = 4$	2.65 ± 0.19	16.56 ± 0.36	3.1
$d = 6$	2.63 ± 0.12	16.48 ± 0.13	3.2

Table F: Ablation study of NA-DARTS.

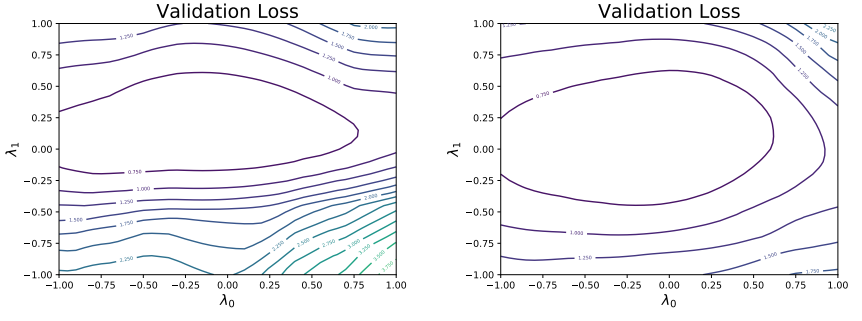
E Loss Landscape Visualization

To qualitatively examine whether our NA-DARTS has found a flat minima, we plot the loss landscape of DARTS and NA-DARTS with the visualization strategy from [11]. Let α denote the architecture found by DARTS or NA-DARTS. We compute the Hessian of the validation loss with respect to α, and v_0 and v_1, which are the eigenvectors corresponding to the two largest eigenvalues of the Hessian matrix. Then we visualize the validation loss of the neighbors of α over the plane spanned by v_0 and v_1. Specifically, we compute the validation loss of the architecture $\alpha + \lambda_0 v_0 + \lambda_1 v_0$, where λ_0 and λ_1 are uniformly sampled from $[-1.0, 1.0]$. The loss values are visualized by the contour plots in Figure A. We observe that the curvature of NA-DARTS at $(0,0)$ (the found architecture α) is much flatter than that of DARTS.

We provide details of the neighboring architecture $\alpha' = \alpha + \lambda_0 v_0 + \lambda_1 v_0$, where we overload the plus sign $(+)$ with the additive representation. Recall that α contains a set of variables representing the operation choice for each edge (i, j): $\alpha = \{\alpha^{(i,j)}\}$. The eigenvectors v_0 and v_1 have the same dimension as α and then can be represented as $v_0 = \{v_0^{(i,j)}\}$ and $v_1 = \{v_1^{(i,j)}\}$. Let $q^{(i,j)} = \lambda_0 v_0^{(i,j)} + \lambda_1 v_1^{(i,j)}$. $\alpha'^{(i,j)}$ is then computed using the additive representation in Eq. 6 ($\alpha_k'^{(i,j)} = \frac{\alpha_k^{(i,j)} + q_k^{(i,j)}}{\Sigma_{k=1}^{n}(\alpha_k^{(i,j)} + q_k^{(i,j)})}$). The eigenvectors v_0 and v_1 are normalized so that the scale of the noise vector $q^{(i,j)}$ is controlled only by λ_0 and λ_1. We use the weights of α obtained in the search as an approximation for the weights of the neighbors α'.



(a) DARTS (standard formulation $\min f(\alpha)$).

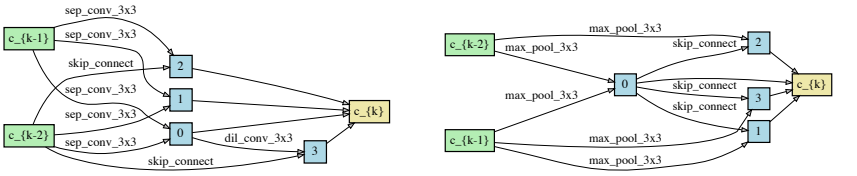


(b) NA-DARTS (neighborhood-aware formulation $\min g\left(f(\mathcal{N}(\alpha))\right)$).

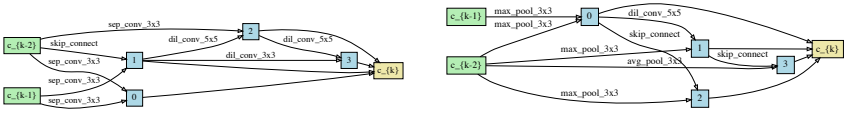
Figure A: Loss landscape visualization of the found architecture. The two plots in Figure Aa (Figure Ab) are generated from two independent runs of DARTS (NA-DARTS). The left plot in Figure Aa and Figure Ab are the same as the plots in Figure 1 in the main text. For the architecture found by DARTS (Figure Aa), we observe that the loss of its neighbors increase drastically as the magnitude of λ_0 or λ_1 increases. However, for the architecture found by our NA-DARTS (Figure Ab), the loss of its neighbors increases much slower. This shows that the architecture found by our NA-DARTS is a much flatter minimum than that found by DARTS.

F Cell Visualization

We visualize the normal cell and reduction cell found by DARTS and our NA-DARTS in Figure B. We observe that the normal cell found by our method NA-DARTS tend to be deeper than that found by DARTS. Normal cells found by our NA-DARTS from different runs have a depth of 3 at most of the time, while normal cells found by DARTS mostly have a depth of 1 or 2. We also observe that the normal cell found by NA-DARTS contains more 5×5 convolution operations. Both of the reduction cells found by DARTS and NA-DARTS contain very few convolution operations. Most operations in the reduction cell do not have parameters, *e.g.*, pooling and skip-connection.



(a) DARTS normal cell (left) and reduction cell (right).



(b) NA-DARTS normal cell (left) and reduction cell (right).

Figure B: Cell Visualization.

References

[1] Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys. In *ICLR*, 2017.

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

[3] John M Danskin. *The theory of max-min and its application to weapons allocation problems*. Springer, 1967.

[4] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[5] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *ICML*, 2017.

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

[7] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *JMLR*, 2019.

[8] Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural Computation*, 1997.

[9] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *ICLR*, 2017.

[10] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[11] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *NeurIPS*, 2018.

[12] Mengtian Li, Ersin Yumer, and Deva Ramanan. Budgeted training: Rethinking deep neural network training under resource constraints. In *ICLR*, 2020.

[13] 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.

[14] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *ICLR*, 2019.

[15] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

[16] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 2015.

[17] Zhewei Yao, Amir Gholami, Qi Lei, Kurt Keutzer, and Michael W Mahoney. Hessian-based analysis of large batch training and robustness to adversaries. In *NeurIPS*, 2018.

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