Neighborhood-Aware Neural Architecture Search

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

Abstract

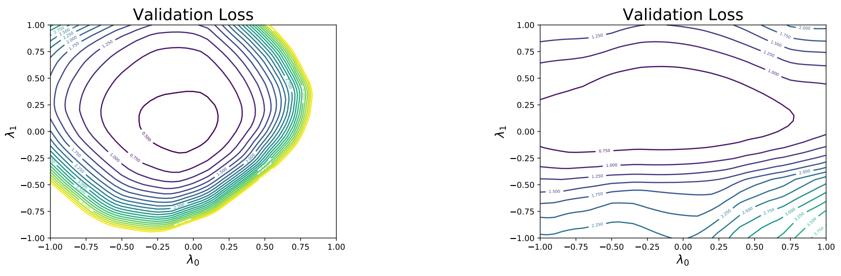
Existing neural architecture search (NAS) methods often return an architecture with good search performance but generalizes poorly to the test setting. To achieve better generalization, we propose a novel neighborhood-aware NAS formulation to identify flat-minima architectures in the search space, with the assumption that flat minima generalize better than sharp minima. The phrase "flat-minima architecture" refers to architectures whose performance is stable under small perturbations in the architecture (*e.g.*, replacing a convolution with a skip connection). Our formulation takes the "flatness" of an architecture into account by aggregating the performance over the neighborhood of this architecture. We demonstrate a principled way to apply our formulation to existing search algorithms, including sampling-based algorithms and gradient-based algorithms. To facilitate the application to gradient-based algorithms, we also propose a differentiable representation for the neighborhood of architectures. Based on our formulation, we propose neighborhood-aware random search (NA-RS) and neighborhood-aware differentiable architecture search (NA-DARTS). Notably, by simply augmenting DARTS with our formulation, NA-DARTS outperforms DARTS and achieves state-of-the-art performance on established benchmarks, including CIFAR-10, CIFAR-100 and ImageNet.

1 Introduction

The process of automatic neural architecture design — neural architecture search (NAS), is a promising technology to improve performance for deep learning applications [17, 29, 30]. NAS methods typically minimize the validation loss to find the optimal architecture. However, directly optimizing such an objective may cause the search algorithm to overfit to the search setting, *i.e.*, finding a solution architecture with good search performance but generalizes poorly to the test setting. This type of overfitting is a result of the differences between the search and test settings, such as the length of training schedules [29, 30], cross-architecture weight sharing [17, 18], and using proxy datasets during search [17, 29, 30].

To achieve better generalization, we propose a novel NAS formulation that searches for "flat-minima architectures", which we define as architectures that perform well under small perturbations of the architecture (Figure 1). One example of architectural perturbations is to replace a convolutional operator with a skip connection (identity mapping). Our work takes inspiration from prior work on neural network training [10], which shows that flat minima

* indicates equal contribution.



(a) Standard formulation (b) Neighborhood-aware formulation

Figure 1: Loss landscape visualization of the found architecture. We project architectures (instead of the network weights) onto a 2D plane. The architectures are sampled along two prominent directions (the two axes, λ_0 and λ_1), with $(0,0)$ denotes the found architecture. We see that our found architecture (right) is a much flatter minimum than that found with the standard formulation (left). We provide visualization details in supplementary materials.

of the loss function correspond to network weights with better generalization than sharp ones. We show that flat minima in the architecture space also generalize better to a new data distribution than sharp minima (Sec. 3.3).

Unlike the standard NAS formulation that directly optimizes single architecture performance, *i.e.*, $\alpha^* = \arg\min_{\alpha \in \mathcal{A}} f(\alpha)$, we optimize the aggregated performance over the neighborhood of an architecture:

$$\alpha^* = \arg\min_{\alpha \in \mathcal{A}} g\left(f(\mathcal{N}(\alpha))\right), \tag{1}$$

where $f(\cdot)$ is a task-specific error metric, α denotes an architecture in the search space $\mathcal{A}$, $\mathcal{N}(\alpha)$ denotes the neighborhood of architecture α, and $g(\cdot)$ is an aggregation function (*e.g.*, the mean function). Note that we overload the notation of the error metric $f(\cdot)$ and define $f(\cdot)$ to return a set of errors when the input is a set of architectures in the neighborhood: $f(\mathcal{N}(\alpha)) = \{f(\alpha') \mid \alpha' \in \mathcal{N}(\alpha)\}$. Common choices for $f(\cdot)$ are validation loss and negative validation accuracy. We will discuss more details of neighborhood $\mathcal{N}(\alpha)$ and aggregation function $g(\cdot)$ in the following text.

To implement our formulation, one must define the neighborhood $\mathcal{N}(\alpha)$ and specify an aggregation function $g(\cdot)$. How to define the neighborhood of an architecture is an open question. One possible method to obtain neighboring architectures is to perturb one or more operations in the architecture and the degree of perturbation defines the scope of the neighborhood. This method can be applied to sampling-based search algorithms, *e.g.*, random search and reinforcement learning. However, it cannot be directly used to generate neighboring architectures for gradient-based search algorithms (*a.k.a*, differentiable NAS), where the neighboring architectures themselves also need to be differentiable with respect to the architecture being learned. To address this issue, we propose a differentiable representation for the neighborhood of architectures, which makes the objective function differentiable and allows us to apply our formulation to gradient-based algorithms, *e.g.*, DARTS [17]. Properly choosing the aggregation function $g(\cdot)$ can help the search algorithm identify flat minima in the search space. Our choice of $g(\cdot)$ (*e.g.*, mean) is inspired by the definition of the flatness/sharpness of local minima in previous work [4, 8, 12].

We summarize our contributions as follows:

1. We propose a neighborhood-aware NAS formulation based on the flat minima assumption, and demonstrate a principled way to apply our formulation to existing search algorithms, including sampling-based algorithms and gradient-based algorithms. We empirically validate our assumption and show that flat-minima architectures generalize better than sharp ones.
2. We propose a neighborhood-aware random search (NA-RS) algorithm and demonstrate its superiority over the standard random search on NAS-Bench-201 [9].
3. We propose a differentiable neighborhood representation so that we can apply our formulation to gradient-based NAS methods. By augmenting DARTS [17] with our formulation, the propoesd algorithm NA-DARTS outperforms DARTS by 1.18% on CIFAR-100 and 1.2% on ImageNet, achieving state-of-the-art performance.

2 Related Work

Flat Minima. Hochreiter & Schmidhuber [10] show that flat minima of the loss function of neural networks generalize better than sharp minima. Flat minima are also used to explain the poor generalization of large-batch methods [12, 26], where large-batch methods are shown to be more likely to converge to sharp minima. Previous work mentioned above focus on flat minima in the network weight space. However, we study flat minima in the architecture space, which is discrete and fundamentally different from the continuous weights studied in previous work. This makes it non-trivial to apply the flat minima idea to NAS.

Zela et al. [27] observes a strong correlation between the generalization error of the architecture found by DARTS [17] and the flatness of the loss function at the found architecture. They propose several regularization strategies to improve DARTS, such as early stopping before the loss curvature becomes too high. Our flat minima assumption is motivated by their observation and our method can be combined with their regularization strategies.

NAS - Search Algorithm. Various search algorithms have been applied to solve NAS, including sampling-based and gradient-based algorithms. Representative sampling-based algorithms include random search [14], reinforcement learning [1, 28, 29, 30], Bayesian optimization [3, 11], evolutionary algorithms [19, 20, 22], and sequential model-based optimization [16]. To make NAS more computationally efficient, weight sharing across architectures is proposed to amortize the training cost of candidate architectures [2, 18]. Based on weight sharing, gradient-based algorithms are proposed to directly learn the architecture with gradient descent [17, 23]. Our focus is not proposing novel search algorithms but revisiting the standard NAS formulation. Our proposed formulation can be applied to both sampling-based algorithms and gradient-based algorithms.

Our work is relevant to SDARTS [5] as both methods take the neighborhood of architectures into consideration. But the goals of the two methods are fundamentally different. SDARTS aims to smooth the loss landscape by finding network weights that are not only good for the current architecture but also the neighborhood of this architecture. However, our goal is to find flat-minima architectures, *i.e.*, finding a solution architecture that not only performs well itself but also has a neighborhood with good performance.

NAS - Search Space. Search space is crucial for the performance of NAS. One of the most widely used search spaces is the cell search space [30], which searches for a cell that can be stacked multiple times to form the entire network. Our proposed neighborhood-aware formulation is agnostic to the search space, and we specifically showcase our formulation on the cell search space.

3 Neighborhood-Aware Formulation

We propose a neighborhood-aware NAS formulation (Eq. 1) to identify flat minima in the search space. Our formulation builds upon the assumption that flat-minima architectures usually generalize better than sharp ones. In this formulation, the optimal architecture is selected according to the aggregated performance $g(f(\mathcal{N}(\alpha)))$ of neighbors of an architecture, instead of the standard criterion, *i.e.*, single architecture performance $f(\alpha)$ only. We now introduce the neighborhood of an architecture $\mathcal{N}(\alpha)$ and the aggregation function $g(\cdot)$.

3.1 Neighborhood Definition and Cell Search Space

Formally defining the neighborhood requires a distance metric between architectures, which largely depends on how an architecture is represented and how the search space is constructed. We adopt the cell search space [30] as it has been widely used in recent NAS methods [16, 17]. Instead of the entire architecture, we search for a cell that can be stacked multiple times to form the entire architecture. The number of times the cell is stacked and the output layer are manually defined prior to the search.

A cell is defined as a directed acyclic graph (DAG) consisting of n nodes. Each node represents a feature map. Each directed edge $(i,j)(1 \leq i < j \leq n)$ is associated with an operation used to transform the feature map at node i, and passes the transformed feature map to node j. The feature map at one node is the sum of all the feature maps on the incoming edges to this node: $x^{(j)} = \sum_{(i,j)\in E} \sum_{k=1}^{m} \alpha_k^{(i,j)} o_k(x^{(i)})$, where E denotes the set of edges in the cell, $x^{(i)}$ is the feature map at node i, and o_k is the k^{th} operation among the m available operations. $\alpha^{(i,j)}$ is a m-dim one-hot vector, indicating the operation choice for edge (i,j). A cell is then represented by a set of variables $\alpha = \{\alpha^{(i,j)}\}$. Note that $\alpha^{(i,j)}$ being a one-hot vector means that only one operation is chosen for edge (i,j). On a side note, the one-hot constraint on $\alpha^{(i,j)}$ can be relaxed in differentiable NAS methods [17, 23].

We define the distance between two cells α and α' as:

$$\text{dist}(\alpha, \alpha') = \sum_{(i,j)\in E} \delta(\alpha^{(i,j)}, \alpha'^{(i,j)}), \tag{2}$$

where $\delta(\cdot,\cdot)$ is the total variation distance between two probability distributions: $\delta(p,q) = \frac{1}{2}||p-q||_1 = \frac{1}{2}\sum_{k=1}^{m} |p_k - q_k|$. Here p and q are both m-dim probability distributions. The total variation distance is symmetric and bounded between 0 and 1. It also offers the following property: $\delta(\alpha^{(i,j)}, \alpha'^{(i,j)}) = 0$ implies that the two cells have the same operation at edge (i,j) and $\delta(\alpha^{(i,j)}, \alpha'^{(i,j)}) = 1$ implies that they have different operations at edge (i,j). Note that instead of directly counting the edge differences, we adopt total variation distance to accommodate relaxed α that is later used in differentiable NAS methods [17, 23].

The neighborhood of a cell α is defined as:

$$\mathcal{N}(\alpha) = \{\alpha' \mid \text{dist}(\alpha, \alpha') \leq d\}, \tag{3}$$

where d is a distance threshold. Due to the property of the total variation distance, when d is an integer, the neighborhood contains all the cells that have at most d edges associated with different operations from α. For clarification, our definition of neighborhood includes the reference architecture α itself.

3.2 Aggregation Function

Given an architecture α, the flatness of its neighborhood 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.

Based on this intuition, we set $g(\cdot)$ as the mean function, since the mean validation loss of architectures around a flat minimum is expected to be lower than those around a sharp minimum. Importantly, minimizing mean$\,(f(\mathcal{N}(\alpha)))$ ensures that α is a local minimum and at the same time has a flat neighborhood. For a similar reason, median and max are also valid choices for $g(\cdot)$ to differentiate between flat minima and sharp minima. We provide more discussions of the aggregation function in supplementary materials.

3.3 Justification of Flat Minima Assumption

3.3.1 Flat Minima Generalize Better

Flat minima in the network weight space are shown to generalize better than sharp ones [10]. However, we focus on flat minima in the architecture space, which is discrete and fundamentally different from the continuous weights studied in previous work. So we conduct experiments to verify that flat minima in the architecture space also generalize better.

NAS-Bench-201 [9] provides a simulated environment for NAS experiments. Using NAS-Bench-201, we search on CIFAR-10 and evaluate the found architectures not only on CIFAR-10, but also on CIFAR-100 and ImageNet-16-120 to better assess the generalization performance of architectures. We select 100 architectures from NAS-Bench-201 that have the lowest validation error on CIFAR-10 to represent local minima in the search space. Next, we show that among these local-minima architectures, flat minima outperform sharp ones, especially on CIFAR-100 and ImageNet-16-120.

We measure the flatness of each local-minimum architecture with its neighborhood variance: the variance of the search-time validation error of its neighboring architectures on CIFAR-10. Based on their neighborhood variance, we divide the 100 architectures into 2 groups: (1) flat minima, which are the 50 architectures with a flat neighborhood (low neighborhood variance), and (2) sharp minima, which are the other 50 architectures with a sharp neighborhood (high neighborhood variance).

We observe that the average search-time validation error of flat minima and sharp minima are almost the same (14.55% and 14.57%). But, as shown in Table 1a, the average test error of flat minima is lower than sharp minima on all three datasets, especially on CIFAR-100 (1.10%) and ImageNet-16-120 (1.24%). This verifies that flat minima generalize better.

3.3.2 Aggregated Performance Gives a Better Ranking of Architectures

Our formulation suggests using the aggregated performance $g\left(f(\mathcal{N}(\alpha))\right)$ as the criterion to select optimal architectures, instead of the standard criterion $f(\alpha)$. The selection criterion determines whether we can obtain an accurate ranking of candidate architectures during search, and further determines the performance of found architectures. So, we evaluate the estimated ranking given by different criteria on NAS-Bench-201 with the Kendall's Tau metric (rank correlation; the higher the better). Table 1b shows that that our criterion $g\left(f(\mathcal{N}(\alpha))\right)$ ($g(\cdot) =$ mean) gives a much more ranking of architectures than the standard criterion $f(\alpha)$. Please see supplementary materials for more details and results.

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

(a)

	CIFAR-10	CIFAR-100	ImageNet-16-120
Baseline	0.66 ± 0.03	0.66 ± 0.02	0.64 ± 0.03
Ours	$\mathbf{0.76 \pm 0.03}$	$\mathbf{0.77 \pm 0.03}$	$\mathbf{0.74 \pm 0.03}$

(b)

Table 1: **(a)**: Average test error of flat-minima architectures and sharp-minima architectures. Flat minima consistently outperform sharp minima on all three datasets. **(b)**: Kendall's Tau (rank correlation) of the standard criterion $f(\alpha)$ (baseline) and our criterion $g(f(\mathcal{N}(\alpha)))$. Our criterion gives a more accurate ranking of architectures on all three datasets.

4 Neighborhood-Aware Search Algorithms

We propose neighborhood-aware search algorithms by applying our formulation to random search (sampling-based) and DARTS (gradient-based), respectively.

4.1 Neighborhood-Aware Random Search (NA-RS)

When applying our formulation to random search, we only need to change the criterion of selecting optimal architectures from $f(\alpha)$ to the aggregated performance $g(f(\mathcal{N}(\alpha)))$. At each step, we randomly sample an architecture α and compute its aggregated performance $g(f(\mathcal{N}(\alpha)))$, and choose the one with the best aggregated performance as our solution. We provide a detailed algorithm sketch of our algorithm NA-RS in supplementary materials.

In practice, the entire neighborhood may be large. Instead of using all the neighbors, we sample a subset of n_{nbr} neighboring architectures from the neighborhood. In our implementation, we always include the reference architecture itself in the sampled subset.

Note that since NA-RS evaluates a neighborhood of architectures at each step, for fair comparison, we allow the standard random search (baseline) to run for more steps such that the two methods evaluate the same number of architectures during search. Specifically, if our NA-RS searches for T steps, the standard random searches for $T \cdot n_{\text{nbr}}$ steps.

While we only present NA-RS, the formulation is also applicable to other sampling-based search algorithms, such as reinforcement learning (RL) and Bayesian optimization (BO). Similar to NA-RS, when applying our formulation to RL or BO, we only need to define the reward signal in RL or the objective function in BO as the aggregated performance $g(f(\mathcal{N}(\alpha)))$. Other components in RL or BO remain unchanged.

4.2 Neighborhood-Aware Differentiable Search

We now present how to apply our formulation to differentiable NAS methods. The key in these methods [6, 17, 23] is to make the objective $f(\alpha)$ differentiable with respect to the architecture α such that one can optimize α with gradient descent.

Similar to the case of random search, our formulation changes the objective from $f(\alpha)$ to $g(f(\mathcal{N}(\alpha)))$. With this change, the differentiability of $g(f(\mathcal{N}(\alpha)))$ is not guaranteed. Therefore, we propose a differentiable neighborhood representation for $\mathcal{N}(\alpha)$ and set the aggregation function $g(\cdot)$ to be mean (g can also be other differentiable functions). This makes $g(f(\mathcal{N}(\alpha)))$ differentiable and allows us to adopt prior gradient estimation techniques, *e.g.*, the continuous relaxation in DARTS [17] or Gumbel-Softmax in SNAS [23], to derive the gradient of $g(f(\mathcal{N}(\alpha)))$. Other parts in the original NAS methods remain the same.

Specifically, we augment DARTS [17] with our formulation and adopt the continuous relaxation in DARTS to estimate the gradient. Therefore, we name our method neighborhood-aware DARTS (NA-DARTS). Note that our formulation is also applicable to other differentiable NAS methods. Specifically, we apply our formulation to DARTS-ES [27] and PC-DARTS [24] and present the results in Table 6.

4.2.1 Neighborhood-Aware DARTS (NA-DARTS)

We first briefly review DARTS and then introduce the formulation of our NA-DARTS.

DARTS. DARTS relaxes the discrete search space to be continuous so that the gradient of the validation loss with respect to the architecture α can be estimated, allowing optimizing α with gradient descent. Concretely, $\alpha^{(i,j)}$ is relaxed from a discrete one-hot vector to a continuous distribution, and is parameterized as the output of the softmax function: $\alpha_k^{(i,j)} = \frac{\exp(\beta_k^{(i,j)})}{\sum_{k=1}^{m}\exp(\beta_k^{(i,j)})}$, where m is the number of available operations and $\beta = \{\beta_k^{(i,j)}\}$ is the set of continuous logits to be learned. DARTS formulates NAS as the following bilevel optimization problem:

$$\min_{\alpha} \mathcal{L}_{\text{val}}(w^*(\alpha), \alpha) \qquad \text{s.t. } w^*(\alpha) = \arg\min_{w} \mathcal{L}_{\text{train}}(w, \alpha), \tag{4}$$

where w denotes network weights, $w^*(\alpha)$ denotes the weights minimizing the training loss of architecture α. $\mathcal{L}_{\text{train}}(w,\alpha)$ and $\mathcal{L}_{\text{val}}(w,\alpha)$ are the training loss and validation loss of architecture α with weights w, respectively.

NA-DARTS. We augment DARTS with our neighborhood-aware formulation:

$$\min_{\alpha} g(\{\mathcal{L}_{\text{val}}(w^*(\alpha'), \alpha') \mid \alpha' \in \mathcal{N}(\alpha)\}) \qquad \text{s.t. } w^*(\alpha') = \arg\min_{w} \mathcal{L}_{\text{train}}(w, \alpha'), \tag{5}$$

where $\mathcal{N}(\alpha)$ is the neighborhood of architecture α and $g(\cdot)$ is an aggregation function.

An outline of our NA-DARTS algorithm can be found in Algorithm 1 in supplementary materials. We first describe how to represent the neighboring architecture α' as a differentiable function of α and, then discuss the gradient estimation for specific choices of $g(\cdot)$.

4.2.2 Differentiable Neighborhood Representation

When the one-hot constraint on α is relaxed, the neighborhood contains an infinite number of neighboring architectures. We propose a method to sample a finite number of architectures from the neighborhood. Importantly, our method allows each sampled neighbor α' to be differentiable with respect to the reference architecture α.

We generate neighboring architectures of α by perturbing the operations associated with the edges in α. We randomly sample d edges to be perturbed from the edges α and leave the operation choice for remaining edges unchanged. This implies that the distance between α and the neighboring architecture α' is at most d, thus as defined in Eq. 3, α' falls into the neighborhood of α. Next, we present how to represent α' as a differentiable function of α.

Let edge (i,j) be an edge to be perturbed. Let $q^{(i,j)}$ be a m-dim real-valued noise vector satisfying the following condition: $|q_k^{(i,j)}| \leq \varepsilon (0 < \varepsilon < 1)$ and $\alpha_k^{(i,j)} + q_k^{(i,j)} \geq 0$ for all $k (1 \leq k \leq m)$. ε is the threshold of the noise. We randomly sample a noise vector $q^{(i,j)}$ and $\alpha'^{(i,j)}$ is computed as:

$$\alpha_k'^{(i,j)} = \frac{\alpha_k^{(i,j)} + q_k^{(i,j)}}{\sum_{k=1}^{n}(\alpha_k^{(i,j)} + q_k^{(i,j)})}. \tag{6}$$

	CIFAR-10	CIFAR-100	ImageNet-16-120
Random Search (RS)	6.39 ± 0.32	29.81 ± 0.44	56.30 ± 1.08
NA-RS (Ours)	$\mathbf{6.20} \pm \mathbf{0.35}$	$\mathbf{28.33} \pm \mathbf{1.22}$	$\mathbf{54.72} \pm \mathbf{0.96}$

Table 2: Test error of NA-RS and the standard random search (RS). NA-RS consistently outperforms RS on all three datasets under the same computational budget.

	Top-1 Test Error (%)			Params (M)	
Method	CIFAR-10	CIFAR-100	ImageNet	CIFAR	ImageNet
DARTS 1st [17]	2.90 ± 0.25	17.66 ± 0.83	-	2.9	-
DARTS 2nd [17]	2.70 ± 0.08	17.72 ± 0.61	26.7	2.9	4.7
NA-DARTS (Ours)	$\mathbf{2.63} \pm \mathbf{0.12}$	$\mathbf{16.48} \pm \mathbf{0.13}$	**25.5**	3.2	4.8

Table 3: Test error of NA-DARTS and DARTS on CIFAR-10, CIFAR-100 and ImageNet. Our NA-DARTS consistently outperforms DARTS on all three datasets.

Repeating the process for each edge to be perturbed will result in a neighboring architecture α', which is differentiable with respect to α. Different noise vectors are sampled for different edges. We term Eq. 6 as the *additive representation* of neighboring architectures.

With the additive representation, we can sample a set of neighboring architectures of α and the sampled architectures are differentiable with respect to α. In practice, we uniformly sample n_{nbr} neighbors from the neighborhood and always include α itself in the sampled set.

4.2.3 Gradient Estimation

After sampling a finite set of neighboring architectures, we compute the validation loss of each individual architecture α', where we use the current weights w as an approximation of $w^*(\alpha')$. Then we pass the set of the validation losses to the aggregation function $g(\cdot)$.

As discussed before, the aggregation function $g(\cdot)$ needs to be differentiable, which immediately rules out median. We choose mean over max due to its superior empirical performance. We note that when using max, Eq. 5 becomes a minimax optimization problem and one can approximate the gradient of the objective using Danskin's Theorem [7]. For completeness, we provide details of using max in supplementary materials.

5 Experimental Results

5.1 Neighborhood-Aware Random Search

We validate our NA-RS on NAS-Bench-201 [9]. Same as the experimental setup in Sec. 3.3, we search on CIFAR-10 and evaluate on CIFAR-10 [13], CIFAR-100 [13], and ImageNet-16-120 [9]. The number of search steps T in NA-RS is set to 100. We set the distance threshold d to 1 and sample 10 neighbors ($n_{\text{nbr}} = 10$) at each step.

As shown in Table 2, NA-RS consistently outperform RS on all three datasets, which validates our neighborhood-aware formulation. Notably, NA-RS outperforms RS by 1.48% on CIFAR-100 and 1.58% ImageNet-16-120. Note that the cell search space typically has a narrow performance range [25], so the improvement brought by our NA-RS is non-trivial. We include the ablation study of n_{nbr} and the aggregation function in NA-RS in supplement.

Method	Test Error (%) CIFAR-10	Test Error (%) CIFAR-100	Params (M)	Search Cost (GPU days)	Search Method
NASNet-A [30]	2.65	17.10*	3.3	1800	RL
AmoebaNet-A [20]	2.84*	17.16*	3.2	3150	Evolution
PNAS [16]	2.95*	17.29*	3.2	225	SMBO
ENAS [18]	2.54*	17.18*	3.9	0.5	RL
SNAS [23]	2.85 ± 0.02	18.25*	2.8	1.5	Gradient
P-DARTS [6]	2.50	16.55	3.4	0.3	Gradient
PC-DARTS [24]	2.57 ± 0.07	16.74*	3.6	0.1	Gradient
DARTS+ [15]	2.72*	16.85*	4.3	0.6	Gradient
SDARTS-ADV [5]	2.61 ± 0.02	16.60*	3.3	1.3	Gradient
DARTS 1st [17]	2.90 ± 0.25	17.66 ± 0.83	2.9	0.3	Gradient
DARTS 2nd [17]	2.70 ± 0.08	17.72 ± 0.61	2.9	1.0	Gradient
NA-DARTS (Ours)	2.63 ± 0.12	$\mathbf{16.48 \pm 0.13}$	3.2	1.1	Gradient

* We train the reported architecture following the training setup in DARTS [17].

Table 4: Comparison with state-of-the-art NAS methods on CIFAR-10 and CIFAR-100. Our NA-DARTS achieves the lowest test error on CIFAR-100. As all the architectures are searched on CIFAR-10, this shows that architectures found by NA-DARTS generalize better.

Method	Test Error (%) Top-1	Test Error (%) Top-5	Params (M)	$+\times$ (M)	Method	Test Error (%) Top-1	Test Error (%) Top-5	Params (M)	$+\times$ (M)
DARTS [17]	26.7	8.7	4.7	574	AmoebaNet-A [20]*	27.0	8.9	5.0	584
P-DARTS [6]*	25.3	8.1	4.9	557	NASNet-A [30]	26.0	8.4	5.3	564
PC-DARTS [24]*	25.7	8.3	5.3	586	ENAS [18]*	26.1	8.6	5.2	576
DARTS+ [15]*	26.4	8.5	5.0	586	PNAS [16]	25.8	8.1	5.1	588
SDARTS-ADV [5]*	25.8	8.5	4.8	545	SNAS [23]	27.3	9.2	4.3	522
NA-DARTS (Ours)	25.5	8.2	4.8	557					

* We train the reported architecture following the training setup in DARTS [17].

Table 5: Comparison with state-of-the-art NAS methods on ImageNet. Our NA-DARTS obtains the second lowest test error on ImageNet.

5.2 Neighborhood-Aware DARTS

DARTS Search Space. Following DARTS [17], we search on CIFAR-10 [13] and evaluate on three datasets: CIFAR-10 [13], CIFAR-100 [13] and ImageNet [21]. The performance on CIFAR-100 and ImageNet are more important, which reflects how well the found architecture can generalize to new datasets. For our NA-DARTS, we sample 10 neighbors at each step, *i.e.*, $n_{\text{nbr}} = 10$. We include more details and ablation results in supplement.

We first compare our NA-DARTS with DARTS. This comparison directly verifies the effectiveness of our neighborhood-aware formulation. As shown in Table 3, NA-DARTS consistently outperforms DARTS on all three datasets. Notably, NA-DARTS outperforms DARTS by 1.18% on CIFAR-100 and 1.2% on ImageNet. Compared with other state-of-the-art NAS methods, NA-DARTS obtains the lowest test error on CIFAR-100 (Table 4) and the second lowest on ImageNet (Table 5).

Note that the cell search space used in DARTS has a narrow performance range [25], *e.g.*, the top-1 error on CIFAR-100 mostly fall around 17%. So the performance gap between our NA-DARTS and DARTS is non-trivial. We also have compared NA-DARTS and DARTS on a different search space and observe a bigger gap (see Table 6).

S3 Search Space. Zela et al. [27] identifies a set of search spaces where DARTS [17] can successfully minimizes the validation loss during search, but the found architectures

	CIFAR-10	CIFAR-100
DARTS [17]	4.13 ± 0.98	22.49 ± 2.62
NA-DARTS (Ours)	$\mathbf{2.97 \pm 0.18}$	$\mathbf{18.86 \pm 0.49}$
DARTS-ES [27]	3.71 ± 1.14	19.21 ± 0.65
NA-DARTS-ES (Ours)	$\mathbf{2.49 \pm 0.02}$	$\mathbf{17.03 \pm 0.41}$
PC-DARTS [24]	$\mathbf{2.66 \pm 0.14}$	17.38 ± 0.45
NA-PC-DARTS (Ours)	2.69 ± 0.08	$\mathbf{16.66 \pm 0.39}$

Table 6: Test error of architectures found from the S3 search space on CIFAR-10 and CIFAR-100. **Top**: Our NA-DARTS significantly outperforms DARTS, *e.g.*, 3.63% on CIFAR-100. **Mid & Bottom**: Applying our formulation to other DARTS extensions, *e.g.*, DARTS-ES and PC-DARTS, can yield further improvement.

are usually degenerated and generalize poorly to the test setting. To further validate our NA-DARTS, we conduct experiments on one search space suggested by Zela et al. [27] and show that in this new search space, NA-DARTS can still generalize much better than DARTS.

The new search space is a subset of the original DARTS search space. The new search space is exactly the same as the original search space, except that it only considers three candidate operations, including 3×3 separable convolution, skip connection, and the zero operation. Following Zela et al. [27], we refer to the new search space as 'S3 search space'.

We search architectures from the S3 search space on CIFAR-10 and then evaluate the found architecture on both CIFAR-10 and CIFAR-100. We see from Table 6 that our NA-DARTS easily outperforms DARTS on both datasets. Notably, NA-DARTS outperforms DARTS by 3.63% on CIFAR-100.

Combining with other DARTS extensions. Many NAS methods like DARTS-ES, P-DARTS and PC-DARTS are all extensions of DARTS and our neighborhood-aware formulation is also applicable to them. Their ideas to improve DARTS, *e.g.*, the early stopping in DARTS-ES or the partial-channel connection idea in PC-DARTS, can be combined with our method for better performance.

To empirically verify this claim, we propose NA-DARTS-ES and NA-PC-DARTS by applying our formulation to DARTS-ES [27] and PC-DARTS [24], respectively. As shown in Table 6, NA-DARTS-ES outperforms DARTS-ES by 1.22% on CIFAR-10 and 2.18% on CIFAR-100. NA-PC-DARTS performs similarly to PC-DARTS on CIFAR-10 but outperforms PC-DARTS by 0.72% on CIFAR-100. As all the architectures are searched on CIFAR-10, the improvement on CIFAR-100 demonstrates that architectures found by our NA-DARTS-ES or NA-PC-DARTS generalize better than DARTS-ES or PC-DARTS.

6 Conclusion

To achieve better generalization, we propose a novel neighborhood-aware NAS formulation, based on the assumption that flat-minima architectures generalize better than sharp ones. Our formulation provides a new perspective for NAS that one should use the aggregated performance over the neighboorhood as the criterion to select optimal architectures. We also demonstrate a principled way to apply our formulation to existing search algorithms and propose two practical search algorithms NA-RS and NA-DARTS. Extensive experiments on CIFAR-10, CIFAR-100 and ImageNet validate the flat minima assumption, and demonstrate the significance of our formulation and algorithms.

Acknowledgement

This project was sponsored by Caterpillar Inc.

References

[1] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. In *ICLR*, 2017.

[2] Gabriel M. Bender, Pieter jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In *ICML*, 2018.

[3] Shengcao Cao, Xiaofang Wang, and Kris M. Kitani. Learnable embedding space for efficient neural architecture compression. In *ICLR*, 2019.

[4] 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.

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

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

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

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

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

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

[11] Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabas Poczos, and Eric P Xing. Neural architecture search with bayesian optimisation and optimal transport. In *NeurIPS*, 2018.

[12] 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.

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

[14] Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. In *UAI*, 2019.

[15] 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.

[16] 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*, 2018.

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

[18] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *ICML*, 2018.

[19] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In *ICML*, 2017.

[20] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *AAAI*, 2019.

[21] 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.

[22] Lingxi Xie and Alan Yuille. Genetic cnn. In *ICCV*, 2017.

[23] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. SNAS: stochastic neural architecture search. In *ICLR*, 2019.

[24] 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.

[25] Antoine Yang, Pedro M. Esperança, and Fabio M. Carlucci. Nas evaluation is frustratingly hard. In *ICLR*, 2020.

[26] 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.

[27] Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. Understanding and robustifying differentiable architecture search. In *ICLR*, 2020.

[28] Zhao Zhong, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. Practical block-wise neural network architecture generation. In *CVPR*, 2018.

[29] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017.

[30] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *CVPR*, 2018.