Stabilizing Differentiable Architecture Search via Perturbation-based Regularization

Xiangning Chen [1] **Cho-Jui Hsieh** [1]

Abstract

Differentiable architecture search (DARTS) is a prevailing NAS solution to identify architectures. Based on the continuous relaxation of the architecture space, DARTS learns a differentiable architecture weight and largely reduces the search cost. However, its stability has been challenged for yielding deteriorating architectures as the search proceeds. We find that the precipitous validation loss landscape, which leads to a dramatic performance drop when distilling the final architecture, is an essential factor that causes instability. Based on this observation, we propose a perturbation-based regularization - SmoothDARTS (SDARTS), to smooth the loss landscape and improve the generalizability of DARTS-based methods. In particular, our new formulations stabilize DARTS-based methods by either random smoothing or adversarial attack. The search trajectory on NAS-Bench-1Shot1 demonstrates the effectiveness of our approach and due to the improved stability, we achieve performance gain across various search spaces on 4 datasets. Furthermore, we mathematically show that SDARTS implicitly regularizes the Hessian norm of the validation loss, which accounts for a smoother loss landscape and improved performance.

1. Introduction

Neural architecture search (NAS) has emerged as a rational next step to automate the trial and error paradigm of architecture design. It is straightforward to search by reinforcement learning (Zoph & Le, 2017; Zoph et al., 2018; Zhong et al., 2018) and evolutionary algorithm (Stanley & Miikkulainen, 2002; Miikkulainen et al., 2019; Real et al., 2017; Liu et al., 2017) due to the discrete nature of the architecture space.

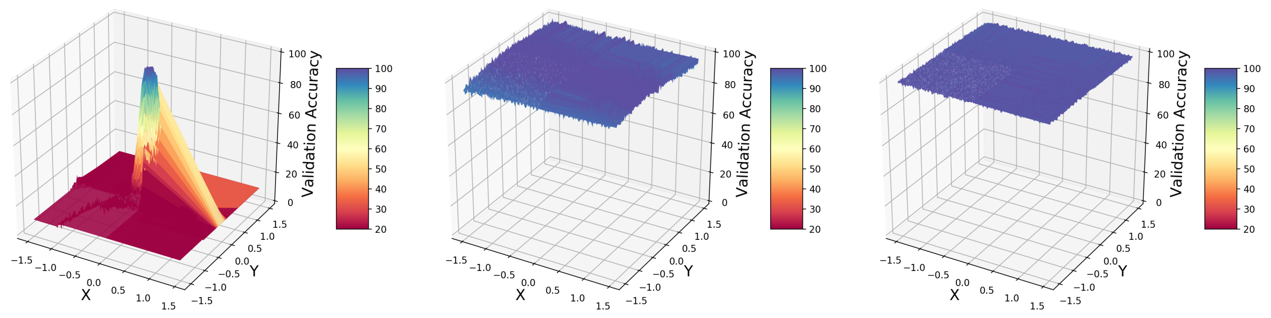
[1]Department of Computer Science, UCLA. Correspondence to: Xiangning Chen <xiangning@cs.ucla.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020. Copyright 2020 by the author(s).

However, these methods usually require massive computation resources. Recently, a variety of approaches are proposed to reduce the search cost including one-shot architecture search (Pham et al., 2018; Bender et al., 2018; Brock et al., 2018), performance estimation (Klein et al., 2017; Bowen Baker, 2018) and network morphisms (Elsken et al., 2019; Cai et al., 2018a;b). For example, one-shot architecture search methods construct a super-network covering all candidate architectures, where sub-networks with shared components also share the corresponding weights. Then the super-network is trained only once, which is much more efficient. Based on this weight-sharing technique, DARTS (Liu et al., 2019) further builds a continuous mixture architecture and relaxes the categorical architecture search problem to learning a differentiable architecture weight A.

Despite being computationally efficient, the stability and generalizability of DARTS have been challenged recently. Many (Zela et al., 2020a; Yu et al., 2020) have observed that although the validation accuracy of the mixture architecture keeps growing, the performance of the derived architecture collapses when evaluation. Such instability makes DARTS converge to distorted architectures. For instance, parameter-free operations such as *skip connection* usually dominate the generated architecture (Zela et al., 2020a), and DARTS has a preference towards wide and shallow structures (Shu et al., 2020). To alleviate this issue, Zela et al. (2020a) propose to early stop the search process based on handcrafted criteria. However, the inherent instability starts from the very beginning and early stopping is a compromise without actually improving the search algorithm.

An important source of such instability is the final projection step to derive the actual discrete architecture from the continuous mixture architecture. There is often a huge performance drop in this projection step, so the validation accuracy of the mixture architecture, which is optimized by DARTS, may not be correlated with the final validation accuracy. As shown in Figure 1(a), DARTS often converges to a sharp region, so small perturbations will dramatically decrease the validation accuracy, let alone the projection step. Moreover, the sharp cone in the landscape illustrates that the network weight w is almost only applicable to the current architecture weight A. Similarly, Bender et al. (2018)



(a) DARTS (b) SDARTS-RS (c) SDARTS-ADV

Figure 1: The landscape of validation accuracy regarding the architecture weight A on CIFAR-10. The X-axis is the gradient direction $\nabla_A L_{valid}$, while the Y-axis is another random orthogonal direction (best viewed in color).

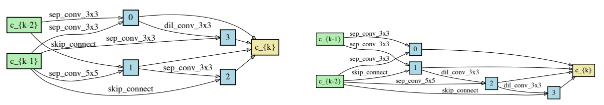
also discovers that the shared weight w of the one-shot network is sensitive and only works for a few sub-networks. This empirically prevents DARTS from fully exploring the architecture space.

To address these problems, we propose two novel formulations. Intuitively, the optimization of A is based on w that performs well on nearby configurations rather than exactly the current one. This leads to smoother landscapes as shown in Figure 1(b, c). Our contributions are as follows:

- We present **SmoothDARTS (SDARTS)** to overcome the instability and lack of generalizability of DARTS. Instead of assuming the shared weight w as the minimizer with respect to the current architecture weight A, we formulate w as the minimizer of the **R**andomly **S**moothed function, defined as the expected loss within the neighborhood of current A. The resulting approach, called SDARTS-RS, requires scarcely additional computational cost but is surprisingly effective. We also propose a stronger formulation that forces w to minimize the worst-case loss around a neighborhood of A, which can be solved by **ADV**ersarial training. The resulting algorithm, called SDARTS-ADV, leads to even better stability and improved performance.

- Mathematically, we show that the performance drop caused by discretization is highly related to the norm of Hessian regarding the architecture weight A, which is also mentioned empirically in (Zela et al., 2020a). Furthermore, we show that both our regularization techniques are implicitly minimizing this term, which explains why our methods can significantly improve DARTS throughout various settings.

- The proposed methods consistently improve DARTS-based methods and can match or improve state-of-the-art results on various search spaces of CIFAR-10, ImageNet, and Penn Treebank. Besides, extensive experiments show that our methods outperform other regularization approaches on 3 datasets across 4 search spaces. Our code is available at https://github.com/xiangning-chen/SmoothDARTS.



(a) SDARTS-RS (b) SDARTS-ADV

Figure 2: Normal cells discovered by SDARTS-RS and SDARTS-ADV on CIFAR-10.

2. Background and Related Work

2.1. Differentiable Architecture Search

Similar to prior work (Zoph et al., 2018), DARTS only searches for the architecture of cells, which are stacked to compose the full network. Within a cell, there are N nodes organized as a DAG (Figure 2), where every node $x^{(i)}$ is a latent representation and every edge (i, j) is associated with a certain operation $o^{(i,j)}$. It is inherently difficult to perform an efficient search since the choice of operation on every edge is discrete. As a solution, DARTS constructs a mixed operation $\bar{o}^{(i,j)}$ on every edge:

$$\bar{o}^{(i,j)}(x) = \sum_{o \in \mathcal{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^{(i,j)})} o(x),$$

where $\mathcal{O}$ is the candidate operation corpus and $\alpha_o^{(i,j)}$ denotes the corresponding architecture weight for operation o on edge (i, j). Therefore, the original categorical choice per edge is parameterized by a vector $\alpha^{(i,j)}$ with dimension $|\mathcal{O}|$. And the architecture search is relaxed to learning a continuous architecture weight $A = [\alpha^{(i,j)}]$. With such relaxation, DARTS formulates a bi-level optimization objective:

$$\min_{A} L_{\text{val}}(w^*(A), A), \text{ s.t. } w^* = \arg\min_{w} L_{\text{train}}(w, A). \quad (1)$$

Then, A and w are updated via gradient descent alternately, where w^* is approximated by the current or one-step forward w. DARTS sets up a wave in the NAS community and many approaches are springing up to make further improvements (Xie et al., 2019; Dong & Yang, 2019; Yao et al., 2020b; Chen et al., 2019; Xu et al., 2020; He et al., 2020; Yao et al., 2020a). For instance, PC-DARTS (Xu et al., 2020) evaluates only a random proportion of channels during search, which can largely reduce the memory overhead. P-DARTS (Chen et al., 2019) attempts to narrow the gap between search and evaluation by progressively increasing the depth of the mixture architecture. Our regularization can be easily applied to these DARTS variants and bring consistent improvements.

Stabilize DARTS. After search, DARTS simply prunes out operations on every edge except the one with the largest architecture weight when evaluation. Under such perturbation, its stability and generalizability have been widely challenged (Zela et al., 2020a; Li & Talwalkar, 2019). Zela et al. (2020a) empirically points out that the dominate eigenvalue λ^A_{max} of the Hessian matrix $\nabla^2_A L_{valid}$ is highly correlated with the stability. They also present an early stopping criterion (DARTS-ES) to prevent λ^A_{max} from exploding. Besides, partial channel connection (Xu et al., 2020), ScheduledDropPath (Zoph et al., 2018) and L2 regularization on w are also shown to improve the stability of DARTS.

NAS-Bench-1Shot1. NAS-Bench-1Shot1 is a benchmark architecture dataset (Zela et al., 2020b) covering 3 search spaces based on CIFAR-10. It provides a mapping between the continuous space of differentiable NAS and discrete space in NAS-Bench-101 (Ying et al., 2019) - the first architecture dataset proposed to lower the entry barrier of NAS. By querying in NAS-Bench-1Shot1, researchers can obtain necessary quantities for a specific architecture (e.g. test accuracy) in milliseconds. Using this benchmark, we track the anytime test error of various NAS algorithms, which allows us to compare their stability.

2.2. Adversarial Robustness

In this paper, we claim that DARTS should be robust against the perturbation on the architecture weight A. Similarly, the topic of adversarial robustness aims to overcome the vulnerability of neural networks against contrived input perturbation (Szegedy et al., 2014). Random smoothing (Lecuyer et al., 2019; Cohen et al., 2019) is a popular method to improve model robustness. Another effective approach is adversarial training (Goodfellow et al., 2015; Madry et al., 2018b), which intuitively optimizes the worst-case training loss. In addition to gaining robustness, adversarial training has also been shown to improve the performance of image classification (Xie et al., 2020) and GAN training (Liu & Hsieh, 2019). To the best of our knowledge, we are the first to apply this idea to stabilize the searching of NAS.

3. Proposed method

3.1. Motivation

During the DARTS search procedure, a continuous architecture weight A is used, but it has to be projected to derive the discrete architecture eventually. There is often a huge performance drop in the projection stage, and thus a good mixture architecture does not imply a good final architecture. Therefore, although DARTS can consistently reduce the validation error of the mixture architecture, the validation error after projection is very unstable and could even blow up, as shown in Figure 3 and 4.

This phenomenon has been discussed in several recent papers (Zela et al., 2020a; Liang et al., 2019), and Zela et al. (2020a) empirically finds that the instability is related to the norm of Hessian $\nabla^2_A L_{\text{valid}}$. To verify this phenomenon, we plot the validation accuracy landscape of DARTS in Figure 1(a), which is extremely sharp – small perturbation on A can hugely reduce the validation accuracy from over 90% to less than 10%. This also undermines DARTS' exploration ability: A can only change slightly at each iteration because the current w only works within a small local region.

3.2. Proposed Formulation

To address this issue, intuitively we want to force the landscape of $L_{\text{val}}(\bar{w}(A), A+\Delta)$ to be more smooth with respect to the perturbation Δ. This leads to the following two versions of SDARTS by redefining $\bar{w}(A)$:

$$\min_{A} L_{\text{val}}(\bar{w}(A), A), \text{ s.t.} \quad (2)$$

$$\text{SDARTS-RS: } \bar{w}(A) = \arg\min_{w} E_{\delta\sim U_{[-\epsilon,\epsilon]}} L_{\text{train}}(w, A+\delta)$$

$$\text{SDARTS-ADV: } \bar{w}(A) = \arg\min_{w} \max_{\|\delta\|\leq\epsilon} L_{\text{train}}(w, A+\delta),$$

where $U_{[-\epsilon,\epsilon]}$ represents the uniform distribution between $-\epsilon$ and ϵ. The main idea is that instead of using w that only performs well on the current A, we replace it by the $\bar{w}$ defined in (2) that performs well within a neighborhood of A. This forces our algorithms to focus on $(\bar{w}, A)$ pairs with smooth loss landscapes. For SDARTS-RS, we set $\bar{w}$ as the minimizer of the expected loss under small random perturbation bounded by ϵ. This is related to the idea of randomized smoothing, which randomly averages the neighborhood of a given function in order to obtain a smoother and robust predictor (Cohen et al., 2019; Lecuyer et al., 2019; Liu et al., 2018b; 2020; Li et al., 2019). On the other hand, we set $\bar{w}$

Algorithm 1 Training of SDARTS

```
Generate a mixed operation $\bar{o}^{(i,j)}$ for every edge $(i, j)$
**while** not converged **do**
    Update architecture $A$ by descending $\nabla_A L_{val}(w, A)$
    Compute $\delta$ based on equation (3) or (4)
    Update weight $w$ by descending $\nabla_w L_{train}(w, A + \delta)$
**end while**
```

to minimize the worst-case training loss under small perturbation of ϵ for SDARTS-ADV. This is based on the idea of adversarial training, which is a widely used technique in adversarial defense (Goodfellow et al., 2015; Madry et al., 2018a).

3.3. Search Algorithms

The optimization algorithm for solving the proposed formulations is described in Algorithm 1. Similar to DARTS, our algorithm is based on alternating minimization between A and w. For SDARTS-RS, $\bar{w}$ is the minimizer of the expected loss altered by a randomly chosen δ, which can be optimized by SGD directly. We sample the following δ and add it to A before running a single step of SGD on w [1]:

$$\delta \sim U_{[-\epsilon,\epsilon]}. \tag{3}$$

This approach is very simple (adding only one line of the code) and efficient (doesn't introduce any overhead), and we find that it is quite effective to improve the stability. As shown in Figure 1(b), the sharp cone disappears and the landscape becomes much smoother, which maintains high validation accuracy under perturbation on A.

For SDARTS-ADV, we consider the worst-case loss under certain perturbation level, which is a stronger requirement than the expected loss in SDARTS-RS. The resulting landscape is even smoother as illustrated in Figure 1(c). In this case, updating $\bar{w}$ needs to solve a min-max optimization problem beforehand. We employ the widely used multi-step projected gradient descent (PGD) on the negative training loss to iteratively compute δ:

$$\delta^{n+1} = \mathcal{P}(\delta^n + lr * \nabla_{\delta^n} L_{\text{train}}(w, A + \delta^n)) \tag{4}$$

where $\mathcal{P}$ denotes the projection onto the chosen norm ball (e.g. clipping in the case of the ℓ_∞ norm) and lr denotes the learning rate.

In the next section, we will mathematically explain why SDARTS-RS and SDARTS-ADV improve the stability and generalizability of DARTS.

[1]We use uniform random for simplicity, while in practice this approach works also with other random perturbations, such as Gaussian.

4. Implicit Regularization on Hessian Matrix

It has been empirically pointed out in (Zela et al., 2020a) that the dominant eigenvalue of $\nabla_A^2 L_{\text{val}}(w, A)$ (spectral norm of Hessian) is highly correlated with the generalization quality of DARTS solutions. In standard DARTS training, the Hessian norm usually blows up, which leads to deteriorating (test) performance of the solutions. In Figure 5, we plot this Hessian norm during the training procedure and find that the proposed methods, including both SDARTS-RS and SDARTS-ADV, consistently reduce the Hessian norms during the training procedure. In the following, we first explain why the spectral norm of Hessian is correlated with the solution quality, and then formally show that our algorithms can implicitly control the Hessian norm.

Why is Hessian norm correlated with solution quality? Assume (w^*, A^*) is the optimal solution of (1) in the continuous space while $\bar{A}$ is the discrete solution by projecting A^* to the simplex. Based on Taylor expansion and assume $\nabla_A L_{\text{val}}(w^*, A^*) = 0$ due to optimality condition, we have

$$L_{\text{val}}(w^*, \bar{A}) = L_{\text{val}}(w^*, A^*) + \frac{1}{2}(\bar{A} - A^*)^T \bar{H}(\bar{A} - A^*), \tag{5}$$

where $\bar{H} = \int_{A^*}^{\bar{A}} \nabla_A^2 L_{\text{val}}(w^*, A) dA$ is the average Hessian. If we assume that Hessian is stable in a local region, then the quantity of $C = \|\nabla_A^2 L_{\text{val}}(w^*, A^*)\| \|\bar{A} - A^*\|^2$ can approximately bound the performance drop when projecting A^* to $\bar{A}$ with a fixed w^*. After fine tuning, $L_{\text{val}}(\bar{w}, \bar{A})$ where $\bar{w}$ is the optimal weight corresponding to $\bar{A}$ is expected to be even smaller than $L_{\text{val}}(w^*, \bar{A})$, if the training and validation losses are highly correlated. Therefore, the performance of $L_{\text{val}}(\bar{w}, \bar{A})$, which is the quantity we care, will also be bounded by C. Note that the bound could be quite loose since it assumes the network weight remains unchanged when switching from A^* to $\bar{A}$. A more precise bound can be computed by viewing $g(A) = L_{\text{val}}(w^*(A), A)$ as a function only paramterized by A, and then calculate its derivative/Hessian by implicit function theory.

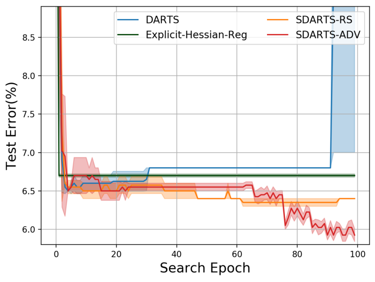


Figure 3: Anytime test error (mean ± std) of DARTS, explicit Hessian regularization, SDARTS-RS and SDARTS-ADV on NAS-Bench-1Shot1 (best viewed in color).

Controlling spectral norm of Hessian is non-trivial. With the observation that the solution quality of DARTS is related to $\|\nabla_A^2 L_{\text{val}}(w^*, A^*)\|$, an immediate thought is to explicitly control this quantity during the optimization procedure. To implement this idea, we add an auxiliary term - the finite difference estimation of Hessian matrix $(\nabla_A L_{\text{val}}(A+\epsilon) - \nabla_A L_{\text{val}}(A-\epsilon))/2\epsilon$ to the loss function when updating A. However, this requires much additional memory to build a computational graph of the gradient, and Figure 3 suggests that it takes some effect compared with DARTS but is worse than both SDARTS-RS and SDARTS-ADV. One potential reason is the high dimensionality – there are too many directions of ϵ to choose from and we can only randomly sample a subset of them at each iteration.

Why can SDARTS-RS implicitly control Hessian? In SDARTS-RS, the objective function becomes

$$\begin{aligned} &E_{\delta\sim U_{[-\epsilon,\epsilon]}} L(w, A+\delta) \\ \approx &E_{\delta\sim U_{[-\epsilon,\epsilon]}} \left[L(w,A) + \delta \nabla_A L(w,A) + \frac{1}{2}\delta^T \nabla_A^2 L(w,A)\delta \right] \quad (6) \\ = &L(w,A) + \frac{\epsilon^2}{6} \text{Tr}\left\{ \nabla_A^2 L(w,A) \right\}, \end{aligned}$$

where the second term in (6) is canceled out since $E[\delta] = 0$ and the off-diagonal elements of the third term becomes 0 after taking the expectation on δ. The update of w in SDARTS-RS can thus implicitly controls the trace norm of $\nabla_A^2 L(w,A)$. If the matrix is close to PSD, this is approximately regularizing the (positive) eigenvalues of $\nabla_A^2 L_{\text{val}}(w,A)$. Therefore, we observe that SDARTS-RS reduces the Hessian norm through its training procedure.

Why can SDARTS-ADV implicitly control Hessian? SDARTS-ADV ensures that the validation loss is small under the worst-case perturbation of A. If we assume the Hessian matrix is roughly constant within ϵ-ball, then adversarial training implicitly minimizes

$$\min_{A:\|A-A^*\|\le\epsilon} L(w,A) \quad (7)$$

$$\approx L(w,A^*) + \frac{1}{2}\max_{\|\Delta\|\le\epsilon} \Delta^T H \Delta \quad (8)$$

when the perturbation is in ℓ_2 norm, the second term becomes the $\frac{1}{2}\epsilon^2\|H\|$, and when the perturbation is in ℓ_∞ norm, the second term is bounded by $\epsilon^2\|H\|$. Thus SDARTS-ADV also approximately minimizes the norm of Hessian. In addition, notice that from (7) to (8) we assume the gradient is 0, which is the property holds only for A^*. In the intermediate steps for a general A, the stability under perturbation will not only be related to Hessian but also gradient, and in SDARTS-ADV we can still implicitly control the landscape to be smooth by minimizing the first-order term in the Taylor expansion of (7).

5. Experiments

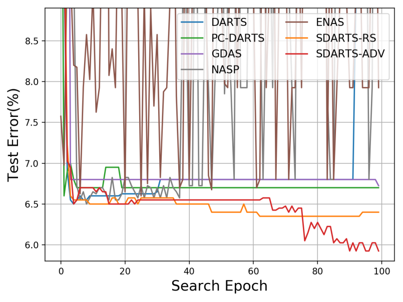
In this section, we first track the anytime performance of our methods on NAS-Bench-1Shot1 in Section 5.1, which demonstrates their superior stability and generalizability. Then we perform experiments on the widely used CNN cell space with CIFAR-10 (Section 5.2) and ImageNet (Section 5.3). We also show our results on RNN cell space with PTB (Section 5.4). In Section 5.5, we present a detailed comparison between our methods and other popular regularization techniques. At last, we examine the generated architectures and illustrate that our methods mitigate the bias for certain operations and connection patterns in Section 5.6.

5.1. Experiments on NAS-Bench-1Shot1

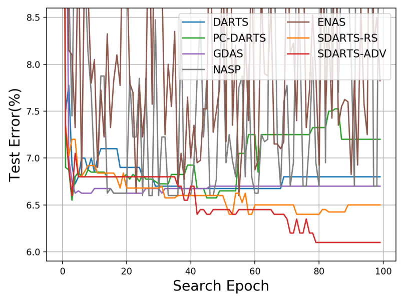
Settings. NAS-Bench-1Shot1 consists of 3 search spaces based on CIFAR-10, which contains 6,240, 29,160 and 363,648 architectures respectively. The macro architecture in all spaces is constructed by 3 stacked blocks, with a *max-pooling* operation in between as the DownSampler. Each block contains 3 stacked cells and the micro architecture of each cell is represented as a DAG. Apart from the operation on every edge, the search algorithm also needs to determine the topology of edges connecting input, output nodes and the choice blocks. We refer to their paper (Zela et al., 2020b) for details of the search spaces.

We make a comparison between our methods with other popular NAS algorithms on all 3 search spaces. Descriptions of the compared baselines can be found in Appendix 7.1. We run every NAS algorithm for 100 epochs (twice of the default DARTS setting) to allow a thorough and comprehensive analysis on search stability and generalizability. Hyperparameter settings for 5 baselines are set as their defaults. For both SDARTS-RS and SDARTS-ADV, the perturbation on A is performed after the softmax layer. We initialize the norm ball ϵ as 0.03 and linearly increase it to 0.3 in all our experiments. The random perturbation δ in SDARTS-RS is sampled uniformly between $-\epsilon$ and ϵ, and we use the 7-step PGD attack under ℓ_∞ norm ball to obtain the δ in SDARTS-ADV. Other settings are the same as DARTS.

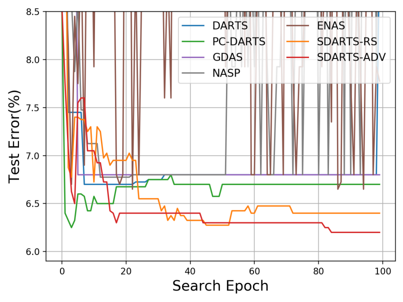
To search for 100 epochs on a single NVIDIA GTX 1080 Ti GPU, ENAS (Pham et al., 2018), DARTS (Liu et al., 2019), GDAS (Dong & Yang, 2019), NASP (Yao et al., 2020b), and PC-DARTS (Xu et al., 2020) require 10.5h, 8h, 4.5h, 5h, and 6h respectively. Extra time of SDARTS-RS is just for the random sample, so its search time is approximately the same with DARTS, which is 8h. SDARTS-ADV needs extra steps of forward and backward propagation to perform the adversarial attack, so it spends 16h. Notice that this can be largely reduced by setting the PGD attack step as 1 (Goodfellow et al., 2015), which brings little performance decrease according to our experiments.



(a) Space 1

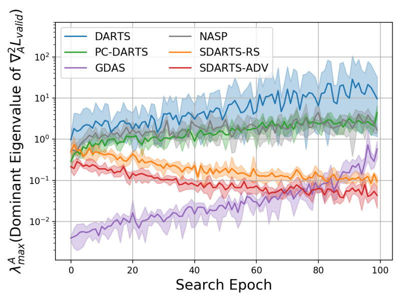


(b) Space 2

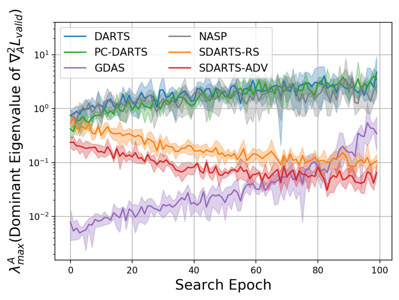


(c) Space 3

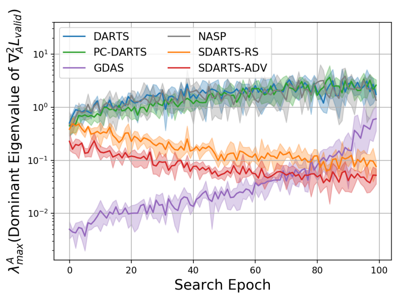
Figure 4: Anytime test error on NAS-Bench-1Shot1 (best viewed in color).



(a) Space 1



(b) Space 2



(c) Space 3

Figure 5: Trajectory (mean $\pm$ std) of the Hessian norm on NAS-Bench-1Shot1 (best viewed in color).

Results. We plot the anytime test error averaged from 6 independent runs in Figure 4. Also, the trajectory (mean $\pm$ std) of the spectral norm of $\nabla^2_A L_{valid}$ is shown in Figure 5. Note that ENAS is not included in Figure 5 since it does not have the architecture weight A. We provide our detailed analysis below.

- DARTS (Liu et al., 2019) generates architectures with deteriorating performance when the search epoch becomes large, which is in accordance with the observations in (Zela et al., 2020a). The single-path modifications (GDAS (Dong & Yang, 2019), NASP (Yao et al., 2020b)) take some effects, e.g. GDAS prevents to find worse architectures and remains stable. However, GDAS suffers premature convergence to sub-optimal architectures, and NASP is only effective for the first few search epochs before its performance starts to fluctuate like ENAS. PC-DARTS is the best baseline on Space 1 and 3, but it also suffers degenerate performance on Space 2.

- SDARTS-RS outperforms all 5 baselines on 3 search spaces. It better explores the architecture space and meanwhile overcomes the instability issue in DARTS. SDARTS-ADV achieves even better performance by forcing w to minimize the worst-case loss around a neighborhood of A. Its anytime test error continues to decrease when the search epoch is larger than 80, which does not occur for any other method.

- As explained in Section 4, the spectral norm λ^A_{max} of Hessian $\nabla^2_A L_{valid}$ has strong correlation with the stability and solution quality. Large λ^A_{max} leads to poor generalizability and stability. In agreement with the theoretical analysis in Section 4, both SDARTS-RS and SDARTS-ADV anneal λ^A_{max} to a low level throughout the search procedure. In comparison, λ^A_{max} in all baselines continue to increase and they even enlarge beyond 10 times after 100 search epochs. Though GDAS (Dong & Yang, 2019) has the lowest λ^A_{max} at the beginning, it suffers the largest growth rate. The partial channel connection introduced in PC-DARTS (Xu et al., 2020) can not regularize the Hessian norm either, thus PC-DARTS has a similar λ^A_{max} trajectory to DARTS and NASP, which match their comparably unstable performance.

5.2. Experiments on CIFAR-10

Settings. We employ SDARTS-RS and SDARTS-ADV to search CNN cells on CIFAR-10 following the search space (with 7 possible operations) in DARTS (Liu et al., 2019).

Table 1: Comparison with state-of-the-art image classifiers on CIFAR-10.

Architecture	**Mean Test Error (%)**	**Params (M)**	**Search Cost (GPU days)**	**Search Method**
DenseNet-BC (Huang et al., 2017)*	3.46	25.6	-	manual
NASNet-A (Zoph et al., 2018)	2.65	3.3	2000	RL
AmoebaNet-A (Real et al., 2019)	3.34 ± 0.06	3.2	3150	evolution
AmoebaNet-B (Real et al., 2019)	2.55 ± 0.05	2.8	3150	evolution
PNAS (Liu et al., 2018a)*	3.41 ± 0.09	3.2	225	SMBO
ENAS (Pham et al., 2018)	2.89	4.6	0.5	RL
NAONet (Luo et al., 2018)	3.53	3.1	0.4	NAO
DARTS (1st) (Liu et al., 2019)	3.00 ± 0.14	3.3	0.4	gradient
DARTS (2nd) (Liu et al., 2019)	2.76 ± 0.09	3.3	1	gradient
SNAS (moderate) (Xie et al., 2019)	2.85 ± 0.02	2.8	1.5	gradient
GDAS (Dong & Yang, 2019)	2.93	3.4	0.3	gradient
BayesNAS (Zhou et al., 2019)	2.81 ± 0.04	3.4	0.2	gradient
ProxylessNAS (Cai et al., 2019)†	2.08	-	4.0	gradient
NASP (Yao et al., 2020b)	2.83 ± 0.09	3.3	0.1	gradient
P-DARTS (Chen et al., 2019)	2.50	3.4	0.3	gradient
PC-DARTS (Xu et al., 2020)	2.57 ± 0.07	3.6	0.1	gradient
R-DARTS(L2) (Zela et al., 2020a)	2.95 ± 0.21	-	1.6	gradient
SDARTS-RS	2.67 ± 0.03	3.4	$0.4^\ddagger$	gradient
SDARTS-ADV	2.61 ± 0.02	3.3	$1.3^\ddagger$	gradient
PC-DARTS-RS	2.54 ± 0.04	3.4	$0.1^\ddagger$	gradient
PC-DARTS-ADV	2.49 ± 0.04	3.5	$0.4^\ddagger$	gradient
P-DARTS-RS	2.50 ± 0.03	3.4	$0.3^\ddagger$	gradient
P-DARTS-ADV	2.48 ± 0.02	3.4	$1.1^\ddagger$	gradient

* Obtained without cutout augmentation.
† Obtained on a different space with PyramidNet (Han et al., 2017) as the backbone.
‡ Recorded on a single GTX 1080Ti GPU.

The macro architecture is obtained by stacking convolution cells for 8 times, and every cell contains $N = 7$ nodes (2 input nodes, 4 intermediate nodes, and 1 output nodes). Other detailed settings for searching and evaluation can be found in Appendix 7.2, which are the same as DARTS (Liu et al., 2019). To further demonstrate the effectiveness of the proposed regularization, we also test our methods on popular DARTS variants PC-DARTS (Xu et al., 2020) (shown as PC-DARTS-RS and PC-DARTS-ADV) and P-DARTS (Chen et al., 2019) (shown as P-DARTS-RS and P-DARTS-ADV).

Results. Table 1 summarizes the comparison of our methods with state-of-the-art algorithms, and the searched normal cells are visualized in Figure 2. Compared with the original DARTS, the random smoothing regularization (SDARTS-RS) decreases the test error from 3.00% to 2.67%, and the adversarial regularization (SDARTS-ADV) further decreases it to 2.61%. When applying to PC-DARTS and P-DARTS, both regularization techniques achieve consistent performance gain and obtain highly competitive results. We also reduces the variance of the search result.

5.3. Experiments on ImageNet

Settings. We test the transferability of our discovered cells on ImageNet. Here the network is constructed by 14 cells and 48 initial channels. We train the network for 250 epochs by an SGD optimizer with an annealing learning rate initialized as 0.5, a momentum of 0.9, and a weight decay of 3×10^{-5}. Similar to previous works (Xu et al., 2020; Chen et al., 2019), we also employ label smoothing and auxiliary loss tower to enhance the training.

Results. As shown in Table 2, both SDARTS-RS and SDARTS-ADV outperform DARTS by a large margin. Moreover, both regularization methods achieve improved accuracy when applying to PC-DARTS and P-DARTS, which demonstrates their generalizability and effectiveness on large-scale tasks. Our best run achieves a top1/5 test error of 24.2%/7.2%, ranking top amongst popular NAS methods.

5.4. Experiments on PTB

Settings. Besides searching for CNN cells, our methods are applicable to various scenarios such as identifying RNN cells. Following DARTS (Liu et al., 2019), the RNN search space based on PTB contains 5 candidate functions, *tanh*, *relu*, *sigmoid*, *identity* and *zero*. The macro architecture of the RNN network is comprised of only a single cell consisting of $N = 12$ nodes. The first intermediate node is manually fixed and the rest nodes are determined by the search algorithm. When searching, we train the RNN network for 50 epochs with sequence length as 35. During

Table 2: Comparison with state-of-the-art image classifiers on ImageNet in the mobile setting.

Architecture	Test Error(%)	
	top-1	top-5
Inception-v1 (Szegedy et al., 2015)	30.1	10.1
MobileNet (Howard et al., 2017)	29.4	10.5
ShuffleNet (v1) (Zhang et al., 2018)	26.4	10.2
ShuffleNet (v2) (Ma et al., 2018)	25.1	10.1
NASNet-A (Zoph et al., 2018)	26.0	8.4
AmoebaNet-C (Real et al., 2019)	24.3	7.6
PNAS (Liu et al., 2018a)	25.8	8.1
MnasNet-92 (Tan et al., 2019)	25.2	8.0
DARTS (Liu et al., 2019)	26.7	8.7
SNAS (mild) (Xie et al., 2019)	27.3	9.2
GDAS (Dong & Yang, 2019)	26.0	8.5
ProxylessNAS (GPU) (Cai et al., 2019)	24.9	7.5
NASP (Yao et al., 2020b)	27.2	9.1
P-DARTS (Chen et al., 2019)	24.4	7.4
PC-DARTS (Xu et al., 2020)	25.1	7.8
SDARTS-RS	25.6	8.2
SDARTS-ADV	25.2	7.8
PC-DARTS-RS	24.7	7.5
PC-DARTS-ADV	24.3	7.4
P-DARTS-RS	24.4	7.4
P-DARTS-ADV	24.2	7.2

evaluation, the final architecture is trained by an SGD optimizer, where the batch size is set as 64 and the learning rate is fixed as 20. These settings are the same as DARTS.

Results. The results are shown in Table 3. SDARTS-RS achieves a validation perplexity of 58.7 and a test perplexity of 56.4. Meanwhile, SDARTS-ADV achieves a validation perplexity of 58.3 and a test perplexity of 56.1. We outperform other NAS methods with similar model size, which demonstrates the effectiveness of our methods for the RNN space. LSTM + SE (Yang et al., 2018) obtains better results than us, but it benefits from a handcrafted ensemble structure.

Table 3: Comparison with state-of-the-art language models on PTB (lower perplexity is better).

Architecture	Perplexity(%)		Params
	valid	test	(M)
LSTM + SE (Yang et al., 2018)*	58.1	56.0	22
NAS (Zoph & Le, 2017)	-	64.0	25
ENAS (Pham et al., 2018)	60.8	58.6	24
DARTS (1st) (Liu et al., 2019)	60.2	57.6	23
DARTS (2nd) (Liu et al., 2019)†	58.1	55.7	23
GDAS (Dong & Yang, 2019)	59.8	57.5	23
NASP (Yao et al., 2020b)	59.9	57.3	23
SDARTS-RS	58.7	56.4	23
SDARTS-ADV	58.3	56.1	23

* LSTM + SE represents LSTM with 15 softmax experts.

† We achieve 58.5 for validation and 56.2 for test when training the architecture found by DARTS (2nd) ourselves.

5.5. Comparison with Other Regularization

Our methods can be viewed as a way to regularize DARTS (implicitly regularize the Hessian norm of validation loss). In this section, we compare SDARTS-RS and SDARTS-ADV with other popular regularization techniques. The compared baselines are 1) partial channel connection (PC-DARTS (Xu et al., 2020)); 2) ScheduledDropPath (Zoph et al., 2018) (R-DARTS(DP)); 3) L2 regularization on w (R-DARTS(L2)); 3) early stopping (DARTS-ES (Zela et al., 2020a)). Descriptions of the compared regularization baselines are shown in Appendix 7.1.

Settings. We perform a thorough comparison on 4 simplified search spaces proposed in (Zela et al., 2020a) across 3 datasets (CIFAR-10, CIFAR-100, and SVHN). All simplified search spaces only contain a portion of candidate operations (details are shown in Appendix 7.3). Following (Zela et al., 2020a), we use 20 cells with 36 initial channels for CIFAR-10, and 8 cells with 16 initial channels for CIFAR-100 and SVHN. The rest settings are the same with Section 5.2. Results in Table 4 are obtained by running every method 4 independent times and pick the final architecture based on the validation accuracy (retrain from scratch for a few epochs).

Results. Our methods achieve consistent performance gain compared with baselines. SDARTS-ADV is the best method for 11 out of 12 benchmarks and we take over both first and second places for 9 benchmarks. In particular, SDARTS-ADV outperforms DARTS, R-DARTS(L2), DARTS-ES, R-DARTS(DP), and PC-DARTS by 31.1%, 11.5%, 11.4%, 10.9%, and 5.3% on average.

5.6. Examine the Searched Architectures

As pointed out in (Zela et al., 2020a; Shu et al., 2020), DARTS tends to fall into distorted architectures that converge faster, which is another manifestation of its instability. So here we examine the generated architectures and see whether our methods can overcome such bias.

5.6.1. Proportion of Parameter-Free Operations

Many (Zela et al., 2020a; Liang et al., 2019) have found out that parameter-free operations such as *skip connection* dominate the generated architecture. Though makes architectures converge faster, excessive parameter-free operations can largely reduce the model's representation capability and bring out low test accuracy. As illustrated in Table 5, we also find similar phenomenon when searching by DARTS on 4 simplified search spaces in Section 5.5. The proportion of parameter-free operations even becomes 100% on S1 and S3, and DARTS can not distinguish the harmful *noise* operation on S4. PC-DARTS achieves some

Table 4: Comparison with popular regularization techniques (test error (%)).
The best method is boldface and underlined while the second best is boldface.

Dataset	Space	DARTS	PC-DARTS	DARTS-ES	R-DARTS(DP)	R-DARTS(L2)	SDARTS-RS	SDARTS-ADV
C10	S1	3.84	3.11	3.01	3.11	**2.78**	**2.78**	**2.73**
	S2	4.85	3.02	3.26	3.48	3.31	**2.75**	**2.65**
	S3	3.34	**2.51**	2.74	2.93	**2.51**	2.53	**2.49**
	S4	7.20	3.02	3.71	3.58	3.56	**2.93**	**2.87**
C100	S1	29.46	24.69	28.37	25.93	24.25	**23.51**	**22.33**
	S2	26.05	22.48	23.25	22.30	22.44	**22.28**	**20.56**
	S3	28.90	21.69	23.73	22.36	23.99	**21.09**	**21.08**
	S4	22.85	21.50	**21.26**	22.18	21.94	21.46	**21.25**
SVHN	S1	4.58	2.47	2.72	2.55	4.79	**2.35**	**2.29**
	S2	3.53	2.42	2.60	2.52	2.51	**2.39**	**2.35**
	S3	3.41	2.41	2.50	2.49	2.48	**2.36**	**2.40**
	S4	3.05	**2.43**	2.51	2.61	2.50	2.46	**2.42**

improvements but is not enough since *noise* still appears. DARTS-ES reveals its effectiveness on S2 and S4 but fails on S3 since all operations found are *skip connection*. We do not show R-DARTS(DP) and R-DARTS(L2) here because their discovered cells are not released. In comparison, both SDARTS-RS and SDARTS-ADV succeed in controlling the portion of parameter-free operations on all search spaces.

5.6.2. Connection Pattern

Shu et al. (2020) demonstrates that DARTS favors wide and shallow cells since they often have smoother loss landscape and faster convergence speed. However, these cells may not generalize better than their narrower and deeper variants (Shu et al., 2020). Follow their definitions (suppose every intermediate node has width c, detailed definitions are shown in Appendix 7.4), the best cell generated by our methods on CNN standard space (Section 5.2) has width $3c$ and depth 4. In contrast, ENAS has width $5c$ and depth 2, DARTS has width $3.5c$ and depth 3, PC-DARTS has width $4c$ and depth 2. Consequently, we succeed in mitigating the bias of connection pattern.

Table 5: Proportion of parameter-free operations in normal cells found on CIFAR-10.

Space	DARTS	PC-DARTS	DARTS-ES	SDARTS-RS	SDARTS-ADV
S1	1.0	0.5	0.375	0.125	0.125
S2	0.875	0.75	0.25	0.375	0.125
S3	1.0	0.125	1.0	0.125	0.125
S4	0.625	0.125	0.0	0.0	0.0

6. Conclusion

We introduce SmoothDARTS (SDARTS), a perturbation-based regularization to improve the stability and generalizability of differentiable architecture search. Specifically, the regularization is carried out with random smoothing or adversarial attack. SDARTS possesses a much smoother landscape and has the theoretical guarantee to regularize the Hessian norm of the validation loss. Extensive experiments illustrate the effectiveness of SDARTS and we outperform various regularization techniques.

Acknowledgement

We thank Xiaocheng Tang, Xuanqing Liu, and Minhao Cheng for the constructive discussions. This work is supported by NSF IIS-1719097, Intel, and Facebook.

References

Bender, G., Kindermans, P.-J., Zoph, B., Vasudevan, V., and Le, Q. Understanding and simplifying one-shot architecture search. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 550–559, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/bender18a.html.

Bowen Baker, Otkrist Gupta, R. R. N. N. Accelerating neural architecture search using performance prediction, 2018. URL https://openreview.net/forum?id=BJypUGZ0Z.

Brock, A., Lim, T., Ritchie, J., and Weston, N. SMASH: One-shot model architecture search through hypernetworks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rydeCEhs-.

Cai, H., Chen, T., Zhang, W., Yu, Y., and Wang, J. Efficient architecture search by network transformation. In *AAAI*, 2018a.

Cai, H., Yang, J., Zhang, W., Han, S., and Yu, Y. Path-level network transformation for efficient architecture search. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 678–687, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018b. PMLR. URL http://proceedings.mlr.press/v80/cai18a.html.

Cai, H., Zhu, L., and Han, S. ProxylessNAS: Direct neural architecture search on target task and hardware. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HylVB3AqYm.

Chen, X., Xie, L., Wu, J., and Tian, Q. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation, 2019.

Cohen, J. M., Rosenfeld, E., and Kolter, J. Z. Certified adversarial robustness via randomized smoothing. In *ICML*, 2019.

DeVries, T. and Taylor, G. W. Improved regularization of convolutional neural networks with cutout, 2017.

Dong, X. and Yang, Y. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1761–1770, 2019.

Elsken, T., Metzen, J. H., and Hutter, F. Efficient multi-objective neural architecture search via lamarckian evolution. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ByME42AqK7.

Goodfellow, I., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. URL http://arxiv.org/abs/1412.6572.

Han, D., Kim, J., and Kim, J. Deep pyramidal residual networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017. doi: 10.1109/cvpr.2017.668. URL http://dx.doi.org/10.1109/CVPR.2017.668.

He, C., Ye, H., Shen, L., and Zhang, T. Milenas: Efficient neural architecture search via mixed-level reformulation, 2020.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.

Huang, G., Liu, Z., Maaten, L. v. d., and Weinberger, K. Q. Densely connected convolutional networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017. doi: 10.1109/cvpr.2017.243. URL http://dx.doi.org/10.1109/CVPR.2017.243.

Klein, A., Falkner, S., Springenberg, J. T., and Hutter, F. Learning curve prediction with bayesian neural networks. In *ICLR*, 2017.

Lecuyer, M., Atlidakis, V., Geambasu, R., Hsu, D., and Jana, S. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 656–672. IEEE, 2019.

Li, B., Chen, C., Wang, W., and Carin, L. Certified adversarial robustness with additive noise. In *Advances in Neural Information Processing Systems*, pp. 9464–9474, 2019.

Li, L. and Talwalkar, A. Random search and reproducibility for neural architecture search, 2019.

Liang, H., Zhang, S., Sun, J., He, X., Huang, W., Zhuang, K., and Li, Z. Darts+: Improved differentiable architecture search with early stopping, 2019.

Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.-J., Fei-Fei, L., Yuille, A., Huang, J., and Murphy, K. Progressive neural architecture search. *Lecture Notes in Computer Science*, pp. 19–35, 2018a. ISSN 1611-3349. doi: 10.1007/978-3-030-01246-5_2. URL http://dx.doi.org/10.1007/978-3-030-01246-5_2.

Liu, H., Simonyan, K., Vinyals, O., Fernando, C., and Kavukcuoglu, K. Hierarchical representations for efficient architecture search, 2017.

Liu, H., Simonyan, K., and Yang, Y. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=S1eYHoC5FX.

Liu, X. and Hsieh, C.-J. Rob-gan: Generator, discriminator, and adversarial attacker. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 11234–11243, 2019.

Liu, X., Cheng, M., Zhang, H., and Hsieh, C.-J. Towards robust neural networks via random self-ensemble. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 369–385, 2018b.

Liu, X., Xiao, T., Si, S., Cao, Q., Kumar, S., and Hsieh, C.-J. How does noise help robustness? explanation and exploration under the neural sde framework. In *CVPR*, 2020.

Luo, R., Tian, F., Qin, T., Chen, E., and Liu, T.-Y. Neural architecture optimization. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 7816–7827. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/8007-neural-architecture-optimization.pdf.

Ma, N., Zhang, X., Zheng, H.-T., and Sun, J. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *The European Conference on Computer Vision (ECCV)*, September 2018.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018a.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018b. URL https://openreview.net/forum?id=rJzIBfZAb.

Miikkulainen, R., Liang, J., Meyerson, E., Rawal, A., Fink, D., Francon, O., Raju, B., Shahrzad, H., Navruzyan, A., Duffy, N., and et al. Evolving deep neural networks. *Artificial Intelligence in the Age of Neural Networks and Brain Computing*, pp. 293–312, 2019. doi: 10.1016/b978-0-12-815480-9.00015-3. URL http://dx.doi.org/10.1016/B978-0-12-815480-9.00015-3.

Pham, H., Guan, M., Zoph, B., Le, Q., and Dean, J. Efficient neural architecture search via parameters sharing. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4095–4104, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/pham18a.html.

Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y. L., Tan, J., Le, Q. V., and Kurakin, A. Large-scale evolution of image classifiers. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 2902–2911. JMLR.org, 2017.

Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. Regularized evolution for image classifier architecture search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:4780–4789, Jul 2019. ISSN 2159-5399. doi: 10.1609/aaai.v33i01.33014780. URL http://dx.doi.org/10.1609/aaai.v33i01.33014780.

Shu, Y., Wang, W., and Cai, S. Understanding architectures learnt by cell-based neural architecture search. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJxH22EKPS.

Stanley, K. O. and Miikkulainen, R. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002. doi: 10.1162/106365602320169811. URL https://doi.org/10.1162/106365602320169811.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014. URL http://arxiv.org/abs/1312.6199.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015. URL http://arxiv.org/abs/1409.4842.

Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., and Le, Q. V. Mnasnet: Platform-aware neural architecture search for mobile. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3–4):229–256, May 1992. ISSN 0885-6125. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

Xie, C., Tan, M., Gong, B., Wang, J., Yuille, A. L., and Le, Q. V. Adversarial examples improve image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 819–828, 2020.

Xie, S., Zheng, H., Liu, C., and Lin, L. SNAS: stochastic neural architecture search. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rylqooRqK7.

Xu, Y., Xie, L., Zhang, X., Chen, X., Qi, G.-J., Tian, Q., and Xiong, H. PC-DARTS: Partial channel connections for memory-efficient architecture search. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJlS634tPr.

Yang, Z., Dai, Z., Salakhutdinov, R., and Cohen, W. W. Breaking the softmax bottleneck: A high-rank RNN language model. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HkwZSG-CZ.

Yao, Q., Chen, X., Kwok, J. T., Li, Y., and Hsieh, C.-J. Efficient neural interaction function search for collaborative filtering. In *Proceedings of The Web Conference 2020*, WWW '20, pp. 1660–1670, New York, NY, USA, 2020a. Association for Computing Machinery. ISBN 9781450370233. doi: 10.1145/3366423.3380237. URL https://doi.org/10.1145/3366423.3380237.

Yao, Q., Xu, J., Tu, W.-W., and Zhu, Z. Efficient neural architecture search via proximal iterations. In *AAAI*, 2020b.

Ying, C., Klein, A., Christiansen, E., Real, E., Murphy, K., and Hutter, F. NAS-bench-101: Towards reproducible neural architecture search. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 7105–7114, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL `http://proceedings.mlr.press/v97/ying19a.html`.

Yu, K., Sciuto, C., Jaggi, M., Musat, C., and Salzmann, M. Evaluating the search phase of neural architecture search. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=H1loF2NFwr`.

Zela, A., Elsken, T., Saikia, T., Marrakchi, Y., Brox, T., and Hutter, F. Understanding and robustifying differentiable architecture search. In *International Conference on Learning Representations*, 2020a. URL `https://openreview.net/forum?id=H1gDNyrKDS`.

Zela, A., Siems, J., and Hutter, F. NAS-BENCH-1SHOT1: Benchmarking and dissecting one-shot neural architecture search. In *International Conference on Learning Representations*, 2020b. URL `https://openreview.net/forum?id=SJx9ngStPH`.

Zhang, X., Zhou, X., Lin, M., and Sun, J. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

Zhong, Z., Yan, J., Wu, W., Shao, J., and Liu, C.-L. Practical block-wise neural network architecture generation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018*, 2018.

Zhou, H., Yang, M., Wang, J., and Pan, W. Bayesnas: A bayesian approach for neural architecture search. In *ICML*, pp. 7603–7613, 2019. URL `http://proceedings.mlr.press/v97/zhou19e.html`.

Zoph, B. and Le, Q. V. Neural architecture search with reinforcement learning. 2017. URL `https://arxiv.org/abs/1611.01578`.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. Learning transferable architectures for scalable image recognition. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun 2018. doi: 10.1109/cvpr.2018.00907. URL `http://dx.doi.org/10.1109/CVPR.2018.00907`.

7. Appendix

7.1. Descriptions of compared baselines

- **ENAS** (Pham et al., 2018) first trains the shared parameter of a one-shot network. For the search phase, it samples sub-networks and use the validation error as the reward signal to update an RNN controller following REINFORCE (Williams, 1992) rule. Finally, they sample several architectures guided by the trained controller and derive the one with the highest validation accuracy.
- **DARTS** (Liu et al., 2019) builds a mixture architecture similar to ENAS. The difference is that it relaxes the discrete architecture space to a continuous and differentiable representation by assigning a weight α to every operation. The network weight w and α are then updated via gradient descent alternately based on the training set and the validation set respectively. For evaluation, DARTS prunes out all operations except the one with the largest α on every edge, which leaves the final architecture.
- **GDAS** (Dong & Yang, 2019) uses the Gumbel-Softmax trick to activate only one operation for every edge during search, similar technique is also applied in SNAS (Xie et al., 2019). This trick reduces the memory cost during search meanwhile keeps the property of differentiability.
- **NASP** (Yao et al., 2020b) is another modification of DARTS via the proximal algorithm. A discrete version of architecture weight $\bar{A}$ is computed every search epoch by applying a proximal operation to the continuous A. Then the gradient of $\bar{A}$ is utilized to update its corresponding A after backpropagation.
- **PC-DARTS** (Xu et al., 2020) evaluates only a random proportion of the channels. This partial channel connection not only accelerates search but also serves as a regularization that controls the bias towards parameter-free operations, as explained by the author.
- **R-DARTS(DP)** (Zela et al., 2020a) runs DARTS with different intensity of ScheduledDropPath regularization (Zoph et al., 2018) and picks the final architecture according to the performance on the validation set. In ScheduledDropPath, each path in the cell is dropped out with a probability that increases linearly over the training procedure.
- **R-DARTS(L2)** (Zela et al., 2020a) runs DARTS with different amounts of L2 regularization and selects the final architecture in the same way with R-DARTS(DP). Specifically, the L2 regularization is applied on the inner loop (i.e. network weight w) of the bi-level optimization problem.
- **DARTS-ES** (Zela et al., 2020a) early stops the search procedure of DARTS if the increase of λ^A_{max} (the dominate eigenvalue of Hessian $\nabla^2_A L_{valid}$) exceeds a threshold. This prevents λ^A_{max}, which is highly correlated with the stability and generalizability of DARTS, from exploding.

7.2. Training details on CNN standard space

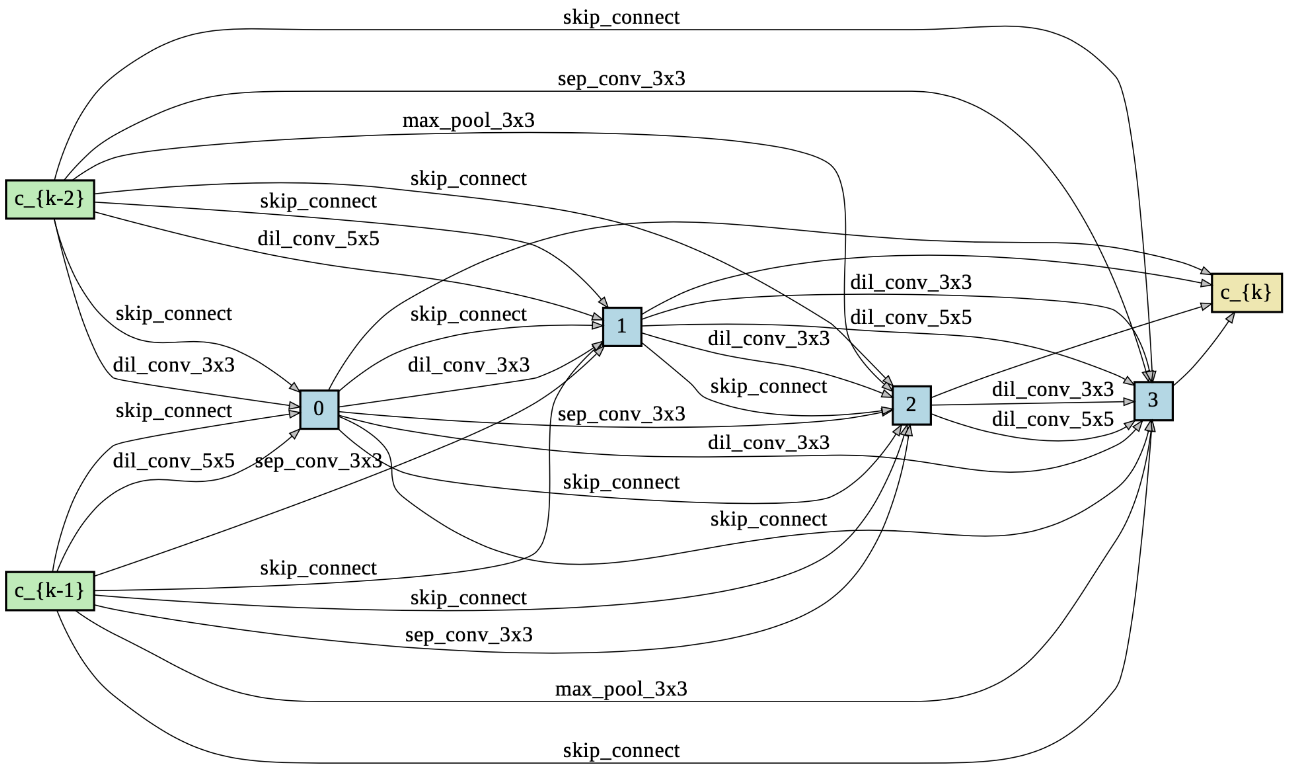
For the search phase, we train the mixture architecture for 50 epochs, with the 50K CIFAR-10 dataset be equally split into training and validation set. Following (Liu et al., 2019), the network weight w is optimized on the training set by an SGD optimizer with momentum as 0.9 and weight decay as 3×10^{-4}, where the learning rate is annealed from 0.025 to 1e-3 following a cosine schedule. Meanwhile, we use an Adam optimizer with learning rate 3e-4 and weight decay 1e-3 to learn the architecture weight A on the validation set. For the evaluation phase, the macro structure consists of 20 cells and the initial number of channels is set as 36. We train the final architecture by 600 epochs using the SGD optimizer with a learning rate cosine scheduled from 0.025 to 0, a momentum of 0.9 and a weight decay of 3e-4. The drop probability of ScheduledDropPath increases linearly from 0 to 0.2, and the auxiliary tower (Zoph & Le, 2017) is employed with a weight of 0.4. We also utilize CutOut (DeVries & Taylor, 2017) as the data augmentation technique and report the result (mean $\pm$ std) of 4 independent runs with different random seeds.

7.3. Micro architecture of 4 simplified search spaces

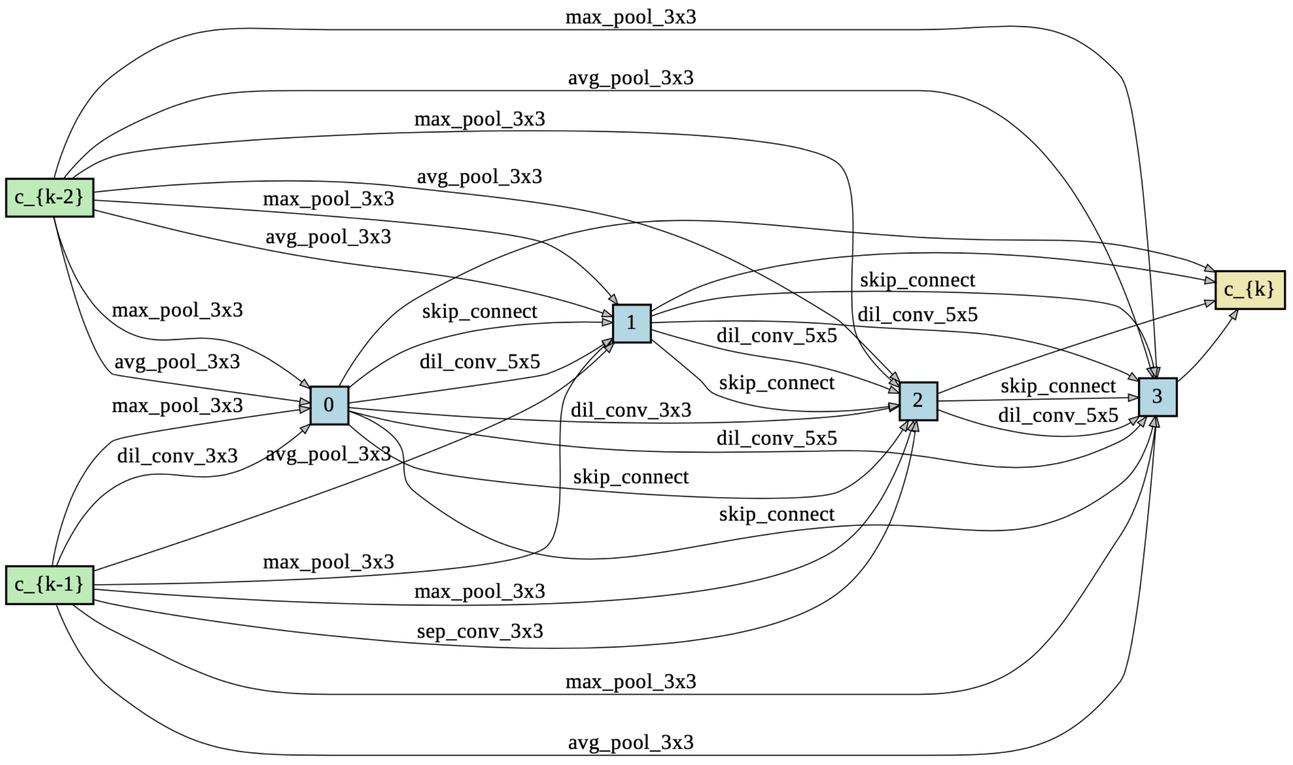
The first space S1 contains 2 popular operators per edge as shown in Figure 6, S2 restricts the set of candidate operations on every edge as $\{3 \times 3$ *separable convolution*, *skip connection*$\}$, the operation set in S3 is $\{3 \times 3$ *separable convolution*, *skip connection*, *zero*$\}$, and S4 simplifies the set as $\{3 \times 3$ *separable convolution*, *noise*$\}$.

7.4. Definitions of cell width and height

Specifically, the depth of a cell is the number of connections on the longest path from input nodes to the output node. While the width of a cell is computed by adding the width of all intermediate nodes that are directly connected to the input nodes, where the width of a node is defined as the channel number for convolutions and the feature dimension for linear operations (In (Shu et al., 2020), they assume the width of every intermediate node is c for simplicity). In particular, if an intermediate node is partially connected to input nodes (i.e. has connections to other intermediate nodes), its width is deducted by the percentage of intermediate nodes it is connected to when computing the cell width.



(a) Normal cell



(b) Reduction cell

Figure 6: Micro cell architecture of S1.