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 and generalizability have 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, named SmoothDARTS (SDARTS), to smooth the loss landscape and improve the generalizability of DARTS. In particular, our new formulations stabilize DARTS 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. The code is available at https://github.com/xiangning-chen/SmoothDARTS.

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. However, these methods usually require massive computation resources. A variety of approaches are then 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. In particular, DARTS (Liu et al., 2019) 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, Chu et al. (2019) and Liang et al. (2019) find that parameter-free operations such as skip connection dominate the generated architecture, and DARTS has a preference towards wide and shallow structures (Shu et al., 2020). To alleviate this issue, some (Zela et al., 2020a; Liang et al., 2019) 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
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 **Randomly Smoothed** 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 **ADVersarial 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 and can match or improve over state-of-the-art results on various search spaces of CIFAR-10 and Penn Treebank. Besides, extensive experiments show that our methods outperform other regularization approaches on three datasets across four search spaces.

![Figure 1](image1.png)

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

![Figure 2](image2.png)

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:

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

where $\mathcal{O}$ is the candidate operation corpus and $\alpha^{(i,j)}_o$ 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:

$$\min_A L_{val}(w^*(A), A), \text{s.t. } w^* = \arg\min_w L_{train}(w, A) \quad (1)$$
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 scenario and many approaches are sprouting up to make further improvements (Xie et al., 2019; Dong & Yang, 2019; Cai et al., 2019; Yao et al., 2020; Xu et al., 2020).

**Stabilize DARTS.** After search, DARTS simply prunes out operations on every edge except the one with the largest $\alpha$ when evaluation. Under such perturbation, its stability and generalizability have been widely challenged (Zela et al., 2020a; Liang et al., 2019; Chu et al., 2019). DARTS+ (Liang et al., 2019) proposes to early stop the search based on the number of skip connection. Zela et al. (2020a) empirically points out that the dominate eigenvalue $\lambda_{max}$ of the Hessian matrix $\nabla^2_w L_{valid}$ is highly correlated with the stability. They also present another early stopping criterion (DARTS-ES) to prevent $\lambda_{max}$ from exploding. Besides, partial channel connection (Xu et al., 2020), Scheduled-DropPath (Zoph et al., 2018) and L2 regularization on $w$ are also shown to improve the stability of DARTS.

**NAS-Bench-1Shot1.** NAS-Bench-1Shot1 (Zela et al., 2020b) is a benchmark architecture dataset covering three search spaces on CIFAR-10. It provides a mapping between the continuous space of differentiable NAS and discrete one 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. 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 $\nabla^2_w L_{valid}$. 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’ ability to explore the architecture space: $A$ 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 $L_{val}(\bar{w}(A), A + \Delta)$ to be more smooth with respect to the perturbation $\Delta$. This leads to the following two versions of SDARTS by redefining $\bar{w}(A)$:

\[
\min_A L_{val}(\bar{w}(A), A), \ s.t. \ (2) \\
SDARTS-RS: \ \bar{w}(A) = \arg \min_w \min_{\|\delta\| \leq \epsilon} E_{\delta \sim U[-\epsilon, \epsilon]} L_{train}(w, A + \delta) \\
SDARTS-ADV: \ \bar{w}(A) = \arg \min_w \max_{\|\delta\| \leq \epsilon} L_{train}(w, A + \delta)
\]

where $U[-\epsilon, \epsilon]$ represents the uniform distribution between $-\epsilon$ and $\epsilon$. 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 $\epsilon$. This is based on the idea of random smoothing, which randomly averaging the neighborhood of a given function to obtain a smoother version (Cohen et al., 2019; Lecuyer et al., 2019). On the other hand, we set $\bar{w}$ to minimize the worst-case training loss under small perturbation of $\epsilon$ for SDARTS-ADV. This is based on the idea of adversarial training, which is a widely used technique in adversarial defense (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 $\delta$, which can be optimized by SGD directly. We sample the following $\delta$ and...
add it to $A$ before running a single step of SGD on $w^1$: 
\[
\delta \sim U_{[-\epsilon, \epsilon]}.
\]
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 $\delta$:
\[
\delta^{n+1} = P(\delta^n + lr \ast \nabla_{\delta^n} L_{train}(w, A + \delta^n))
\]
where $P$ denotes the projection onto the chosen norm ball (e.g. clipping in the case of the $\ell_\infty$ 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.

4. Implicit Regularization on Hessian Matrix

It has been empirically pointed out in (Zela et al., 2020a) that the dominant eigenvalue of $\nabla^2_w L_{val}(w, A)$ (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 $A$ is the discrete solution by projecting $A^*$ to the simplex. Based on Taylor expansion and assume $\nabla_A L_{val}(w^*, A^*) = 0$ due to optimality condition, we have
\[
L_{val}(w^*, A) = L_{val}(w^*, A^*) + \frac{1}{2}(A - A^*)^T \bar{H}(A - A^*)
\]
where $\bar{H} = \int_{A^*} \nabla^2_A L_{val}(w, A)dA$ is the average Hessian. If we assume that Hessian is stable in a local region, then the quantity of $C = ||\nabla^2_A L_{val}(w^*, A^*)|| ||\bar{A} - A^*||^2$ can approximately bound the performance drop when projecting $A^*$ to $A$ with a fixed $w^*$. After fine tuning, $L_{val}(\bar{\bar{w}}, \bar{\bar{A}})$ where $\bar{\bar{w}}$ is the optimal weight corresponding to $A$ is expected to be even smaller than $L_{val}(w^*, A)$, if the training and validation losses are highly correlated. Therefore, the performance of $L_{val}(\bar{\bar{w}}, \bar{\bar{A}})$, 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 $A$. A more precise bound can be computed by viewing $g(A) = L_{val}(w^*(A), A)$ as a function only parameterized by $A$, and then calculate its derivative/Hessian.

Controlling spectral norm of Hessian is non-trivial.

With the observation that the solution quality of DARTS is related to $||\nabla^2_A L_{val}(w^*, A^*)||$, 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 $\nabla_A L_{val}(A + \epsilon) - \nabla_A L_{val}(A - \epsilon)$ 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 $\epsilon$ to choose from and we can only randomly sample a subset of them at each iteration.

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

In SDARTS-RS, the objective function becomes
\[
E_{\delta \sim U_{[-\epsilon, \epsilon]}} L(w, A + \delta) = \frac{E}{6} \left\{ \nabla^2_A L(w, A) \right\}
\]
where the second term in (7) is canceled out since \(E[\delta] = 0\) and the off-diagonal elements of the third term becomes 0 after taking the expectation on \(\delta\). The update of \(w\) in SDARTS-RS can thus implicitly controls the trace norm of \(\nabla^2_A L(w, A)\). If the matrix is close to PSD, this is approximately regularizing the (positive) eigenvalues of \(\nabla^2_A L_{\text{val}}(w, A)\). Therefore, we observe that SDARTS-RS empirically 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 \(\epsilon\)-ball, then adversarial training implicitly minimizes
\[
\min_{A: \|A - A^*\| \leq \epsilon} L(w, A) \tag{9}
\]
\[
\approx L(w, A^*) + \frac{1}{2} \max_{\|\Delta\| \leq \epsilon} \Delta^T H \Delta \tag{10}
\]
when the perturbation is in \(\ell_2\) norm, the second term becomes the \(\frac{1}{2} \epsilon^2 \|H\|\), and when the perturbation is in \(\ell_\infty\) 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 (9) to (10) 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 (9).

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 on CIFAR-10 (Section 5.2) and RNN cell space on PTB (Section 5.3). In Section 5.4, we present a detailed comparison between our methods with other popular regularization techniques. At last, we examine the generated architectures and illustrate that our methods mitigate DARTS’ bias for certain operations and connection patterns in Section 5.5.

5.1. Architecture Search 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 of models 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. Besides 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 about the search spaces.

We make a comparison between our methods and state-of-the-art 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 default. For both SDARTS-RS and SDARTS-ADV, the perturbation on \(A\) is performed after the softmax layer. We initialize the norm ball \(\epsilon\) as 0.03 and linearly increase it to 0.3 in all our experiments. The random perturbation \(\delta\) in SDARTS-RS is sampled uniformly between \(-\epsilon\) and \(\epsilon\). And we use the 7-step PGD attack under \(\ell_\infty\) norm ball to obtain the \(\delta\) in SDARTS-ADV. Other settings are the same as DARTS.

To search for 100 epochs on a single NVIDIA GTX 1080 Ti GPU, ENAS, DARTS, GDAS, NASP, PC-DARTS requires 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 as 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 (FGSM (Goodfellow et al., 2015)), which only brings little performance decrease according to our experiments.

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 \(\nabla^2_{\text{val}} L\) is shown in Figure 5. Noting that ENAS is not included in Figure 5 since it does not have the architecture weight \(A\). We provide our detailed analysis below.

- DARTS generates architectures with deteriorating performance when the search epoch becomes large, which is in accordance with the observations in (Zela et al., 2020a; Liang et al., 2019). The single-path modifications (GDAS, NASP) take effects to some extent, e.g. GDAS prevents to find worse architectures and remains stable. However, GDAS suffers premature convergence to sub-optimal architectures, and NASP
Stabilizing Differentiable Architecture Search via Perturbation-based Regularization

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

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

is effective for the first few search epochs before its performance starts to fluctuate like ENAS. A potential reason is that the architecture weight $A$ is clipped to the nearest boundary when it cannot satisfy some range constraint. This makes NASP confused when choosing among operations if their corresponding weights are similar on certain edges. The partial channel connection introduced by PC-DARTS makes it the best baseline on Space 1 and 3, but PC-DARTS also suffers severely 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 $\lambda_{max}^A$ of Hessian $\nabla^2_{A} L_{\text{valid}}$ has strong correlation with the stability and solution quality. Large $\lambda_{max}^A$ leads to poor generalizability and stability. In agreement with the theoretical analysis that our methods keep minimizing $\lambda_{max}^A$ (Section 4), both SDARTS-RS and SDARTS-ADV anneal $\lambda_{max}^A$ to a low level throughout the search procedure. In comparison, $\lambda_{max}^A$ in all baselines continue to increase and they even enlarge beyond 10 times after 100 search epochs. Though GDAS has the lowest $\lambda_{max}^A$ at the beginning, it suffers the largest growth rate. The partial channel connection in PC-DARTS can not regularize the Hessian norm, it has a similar $\lambda_{max}^A$ trajectory to DARTS and NASP, which supports their comparably unstable performance.

5.2. Architecture Search on CNN Standard Space

Settings. We employ SDARTS-RS and SDARTS-ADV to search CNN cells on CIFAR-10 following the search space (with 7 operations) in DARTS (Liu et al., 2019). 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 node). Other detailed settings for searching and evaluation can be found in Appendix 7.2, which are the same as DARTS.

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. We achieve performance gain compared with DARTS and most of its variants. Moreover, the variance of SDARTS-RS is considerably bet-
15 epochs, and the search epoch is comparably smaller, which may alleviate its instability issue discussed in Section 5.1. Nevertheless, when searching on various simplified search spaces across 3 datasets, our methods achieve superior stability and test accuracy compared with PC-DARTS as indicated in Section 5.4.

### 5.3. Architecture Search on RNN Standard Space

**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, i.e. 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 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 2. 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 obtains better results than us, but it benefits from a handcrafted ensemble structure.

| Architecture            | 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., 2018)* | 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., 2020) | 2.83 ± 0.09   | 3.3        | 0.1                    | 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*                   | gradient        |
| SDARTS-ADV                 | 2.61 ± 0.02   | 3.3        | 1.3†                   | 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.

### 5.4. 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.

| Architecture            | Perplexity(%) | Params (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., 2020) | 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.
Stabilizing Differentiable Architecture Search via Perturbation-based Regularization

Table 3: 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        | 2.75       | 2.75       |
|         | 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 | 18.87    | 28.37    | 25.93       | 24.25       | 17.82      | 16.88      |
|         | S2    | 26.05 | 18.23    | 23.25    | 22.30       | 22.44       | 17.56      | 17.24      |
|         | S3    | 28.90 | 18.05    | 23.73    | 22.36       | 23.99       | 17.73      | 17.12      |
|         | S4    | 22.85 | 17.16    | 21.26    | 22.18       | 21.94       | 17.17      | 15.46      |
| SVHN    | S1    | 4.58  | 2.28     | 2.72     | 2.55        | 4.79        | 2.26       | 2.16       |
|         | S2    | 3.53  | 2.39     | 2.60     | 2.52        | 2.51        | 2.37       | 2.37       |
|         | S3    | 3.41  | 2.27     | 2.50     | 2.49        | 2.48        | 2.21       | 2.05       |
|         | S4    | 3.05  | 2.37     | 2.51     | 2.61        | 2.50        | 2.35       | 1.98       |

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 search spaces utilize the same macro architecture as in Section 5.2, the difference is that they only contain a portion of candidate operations (details are shown in Appendix 7.3). Results in Table 3 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). Other settings are the same as Section 5.2.

Results. The discovered cells are shown in Appendix (Figure 7, 8, 9 and 10). Our methods achieve substantial performance gains compared with baselines. SDARTS-ADV is the best method for all 12 benchmarks and SDARTS-RS strikes the second place on 10 benchmarks. The cell discovered on S3 for CIFAR-10 even achieves higher test accuracy than all the methods in Table 1 (except for ProxylessNAS that searches based on PyramidNet).

5.5. Examine the Searched Architectures

As pointed out in (Zela et al., 2020a; Liang et al., 2019; 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.

Table 4: 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        |

5.5.1. 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 capacity and bring out low test accuracy. As illustrated in Table 4, we also find similar phenomenon when searching by DARTS on 4 simplified search spaces in Section 5.4. 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 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.

5.5.2. Connection Pattern

Shu et al. (2020) demonstrates, from both empirical and theoretical aspects, that DARTS tends to favor 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.

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.
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, Stockholmsmssan, 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, Stockholmsmssan, 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=Hy1VB3AqYm.

Chu, X., Zhou, T., Zhang, B., and Li, J. Fair darts: Eliminating unfair advantages in differentiable architecture search, 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.

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.

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. 1935, 2018. 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.

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.,
Stabilizing Differentiable Architecture Search via Perturbation-based Regularization

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., Dufy, N., and et al. Evolving deep neural networks. *Artificial Intelligence in the Age of Neural Networks and Brain Computing*, pp. 293-312, 2019. doi: 10.1007/B978-0-12-815480-9.00015-3. URL http://dx.doi.org/10.1007/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, Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/pham18a.html.

Rea, 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, ICML17, 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=BJxH2ZEKPS.

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.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(34):229256, May 1992. ISSN 0885-6125. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

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=BJ1S634tPr.

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

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.

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

- **NASP** (Yao et al., 2020) is another modification of DARTS via the proximal algorithm. A discrete version of architecture weight $\alpha$ is computed every search epoch by applying a proximal operation to the continuous $A$. Then the gradient of $A$ 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 $\lambda_{\text{max}}^A$ (the dominate eigenvalue of Hessian $\nabla_A^2L_{\text{valid}}$) exceeds a threshold. This prevents $\lambda_{\text{max}}^A$, 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 $3 \times 10^{-4}$, where the learning rate is annealed from 0.25 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 ± 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 $\{3 \times 3 \text{ separable convolution}, \text{skip connection}\}$, the operation set in S3 is $\{3 \times 3 \text{ separable convolution}, \text{skip connection, zero}\}$, and S4 simplifies the set as $\{3 \times 3 \text{ separable convolution, noise}\}$.

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 deduced by the percentage of intermediate nodes it is connected to when computing the cell width.
Figure 6: Micro cell architecture of S1.
Figure 7: Normal cells discovered by SDARTS-RS on spaces S1-S4 across CIFAR-10, CIFAR-100 and SVHN.

Figure 8: Reduction cells discovered by SDARTS-RS on spaces S1-S4 across CIFAR-10, CIFAR-100 and SVHN.
Figure 9: Normal cells discovered by SDARTS-ADV on spaces S1-S4 across CIFAR-10, CIFAR-100 and SVHN.

Figure 10: Reduction cells discovered by SDARTS-ADV on spaces S1-S4 across CIFAR-10, CIFAR-100 and SVHN.