UNDERSTANDING ARCHITECTURES LEARNT BY CELL-BASED NEURAL ARCHITECTURE SEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural architecture search (NAS) searches architectures automatically for given tasks, e.g., image classification and language modeling. Improving the search efficiency and effectiveness have attracted increasing attention in recent years. However, few efforts have been devoted to understanding the generated architectures and particularly the commonality these architectures may share. In this paper, we first summarize the common connection pattern of NAS architectures. We empirically and theoretically show that the common connection pattern contributes to a smooth loss landscape and more accurate gradient information, and therefore fast convergence. As a consequence, NAS algorithms tend to select architectures with such common connection pattern during architecture search. However, we show that the selected architectures with the common connection pattern may not necessarily lead to a better generalization performance compared with other candidate architectures in the same search space, and therefore further improvement is possible by revising existing NAS algorithms.

1 INTRODUCTION

Over the past decades, various neural network architectures (Krizhevsky et al., 2012; Simonyan & Zisserman, 2015; He et al., 2016; Huang et al., 2017) have been proposed, achieving superhuman performance for a wide range of tasks. Designing these neural networks typically takes substantial efforts from domain experts by trial and error. Recently, there is a growing interest in neural architecture search (NAS), which automatically searches for high-performance architectures for the given task. The searched NAS architectures (Zoph et al., 2018; Real et al., 2018; Pham et al., 2018; Liu et al., 2018; Xie et al., 2018; Luo et al., 2018; Cai et al., 2019; Akimoto et al., 2019; Nayman et al., 2019) have outperformed best expert-designed architectures on many computer vision and natural language processing tasks.

Mainstream NAS algorithms typically search for the connection topology and transforming operation accompanying each connection from a predefined search space. Tremendous efforts have been exerted to develop efficient and effective NAS algorithms (Liu et al., 2018; Xie et al., 2018; Luo et al., 2018; Akimoto et al., 2019; Nayman et al., 2019). However, less attention has been paid to these searched architectures for further insight. To our best knowledge, there is no related work in the literature examining whether these NAS architectures share any pattern, and why the pattern matters for the architecture search. The two questions are fundamental to understand and improve existing NAS algorithms. In this paper, we endeavor to address the questions by examining the popular NAS architectures[1].

The recent work (Xie et al., 2019) shows that the architectures with random connection typologies can achieve state-of-the-art accuracy on various datasets. Inspired by this result, we examine the connection pattern of the architectures generated by popular NAS algorithms. In particular, we find a common connection pattern of the popular NAS architectures that these architectures favor shallow and wide cells, where the majority of the intermediate nodes are connected to the input nodes.

To appreciate this particular connection pattern, we first visualize the training process of the popular NAS architectures and their randomly connected variants. A faster and more stable convergence

[1]The popular NAS architectures refer to the best architectures generated by state-of-the-art (SOTA) NAS algorithms throughout the paper. Notably, we focus on the NAS architectures of repeating cells.

is observed for the architectures with common connection pattern, leading to the selection of these architectures during the search. We further empirically and theoretically show that the popular NAS architectures with the common connection pattern consistently achieve a smoother loss landscape and smaller gradient variance than their random variants, which explains the better convergence behavior and thus the selection of these NAS architectures.

We finally evaluate the generalization performance of the popular NAS architectures. Although the architectures with the common connection pattern consistently converge faster, they may not generalize better than other candidate architectures. We therefore believe that re-thinking NAS from the perspective of generalization rather than convergence of candidate architectures should potentially help obtain significantly better architectures.

2 Related Works

Neural Architecture Search Neural architecture search (NAS) searches for best-performing architectures automatically for a given task. It has received increasing attention in recent years due to its outstanding performance and the demand for automated machine learning (AutoML). There are three major components in NAS as summarized by Elsken et al. (2019), namely search space, search policy (or strategy, algorithm), and performance evaluation (or estimation). The prior knowledge extracted from expert-designed architectures is exploited to define the search space. Different search policies are proposed to improve the search effectiveness (Zoph et al., 2018; Real et al., 2018; Pham et al., 2018; Cai et al., 2019) and the search efficiency (Liu et al., 2018; Xie et al., 2018; Luo et al., 2018; Nayman et al., 2019; Akimoto et al., 2019). Some search policies (Tan et al., 2018) also consider the inference time of the generated architectures. However, few efforts have been devoted to understanding the best architectures generated by various NAS approaches. Detailed analysis of these architectures could give further insights into what the NAS algorithms are learning and why they prefer these specific architectures.

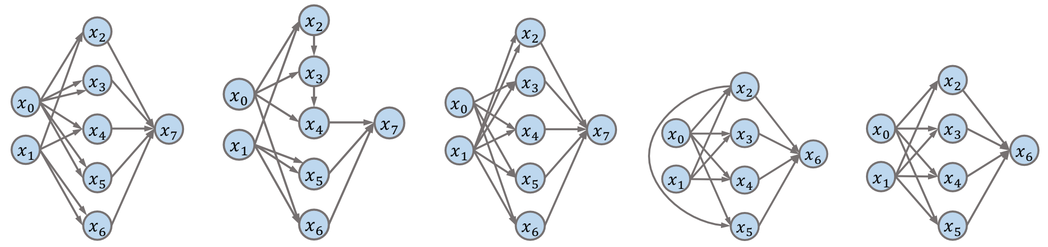
Evaluation of NAS algorithms Recent works evaluate NAS algorithms comparing with random search methods. Li & Talwalkar (2019) and Sciuto et al. (2019) compare the generalization performance of architectures generated from random search and existing NAS algorithms. Interestingly, the random search can find architectures with comparable or even better generalization performance. Sciuto et al. (2019) empirically shows that the ineffectiveness of some NAS algorithms (Zoph et al., 2018; Pham et al., 2018) could be the consequence of the weight sharing mechanism during architecture search. Similarly, Xie et al. (2019) generates several architectures with random connections and identical operations, which achieves SOTA performance. While these evaluations on NAS algorithms help understand their disadvantages, what they are learning and why they learn these specific architectures are still not well understood.

3 The Common Connection Pattern of Popular NAS Cells

Mainstream NAS algorithms (Zoph et al., 2018; Real et al., 2018; Pham et al., 2018; Liu et al., 2018; Xie et al., 2018; Luo et al., 2018) typically search for the cell structure, including the connection topology and the corresponding operation (transformation) coupling each connection. The generated cell is then replicated to construct the entire neural network. We therefore mainly focus on the study of these cell-based NAS architectures. In this section, we first introduce the commonly adopted cell representation, which is useful to understand the connection and computation in a cell space. We then sketch the connection topology to investigate the connection pattern of cell-based NAS architecture. By the comparison, we observe that there is a common connection pattern among cells learned by different NAS algorithms, indicating that those cells tend to be wide and shallow.

3.1 Cell Representation

Following DARTS (Liu et al., 2018), we represent the cell topology as a directed acyclic graph (DAG) consisting of N nodes, including M input nodes, one output node and $(N - M - 1)$ intermediate nodes. Each node forms a latent representation of the input instance. The input nodes are the outputs from M preceding cells. And the output node aggregates (e.g., concatenate) the representations from all intermediate nodes. Moreover, each intermediate node is connected to M



(a) NASNet, $5c$, 2 (b) AmoebaNet, $3c$, 4 (c) ENAS, $5c$, 2 (d) DARTS, $3c$, 3 (e) SNAS, $4c$, 2

Figure 1: Topologies of cells from popular NAS architectures. Each sub-figure has three sets of nodes from left to right, which are the input nodes, intermediate nodes, and output node respectively. The arrows, i.e. operations of the cell, represent the direction of information flow. The caption of each sub-figure includes the name of the architecture, width and depth of the cell following our definition. The width of a cell is calculated by assuming that all intermediate nodes share the same width c.

proceeding nodes in the same cell. Each connection transforms the representation from a node via an operation from a predefined operation set, including 3×3 convolution, 3×3 max pooling, etc. The target of NAS algorithm is to search for the best M source nodes that each intermediate node is connected to and the operation coupling each connection. In the literature, this searched cell is then replicated by L times to build the entire neural network architecture[2].

We abuse the notation C to denote a cell and the neural network architecture built with the cell in the following sections. Besides, we shall use C^A to denote the best architecture (or cell) searched from the NAS algorithm A (i.e., DARTS (Liu et al., 2018), ENAS (Pham et al., 2018), etc.). The details on how to build the entire architectures with given cells are described in Appendix A.3.

3.2 The Common Connection Pattern

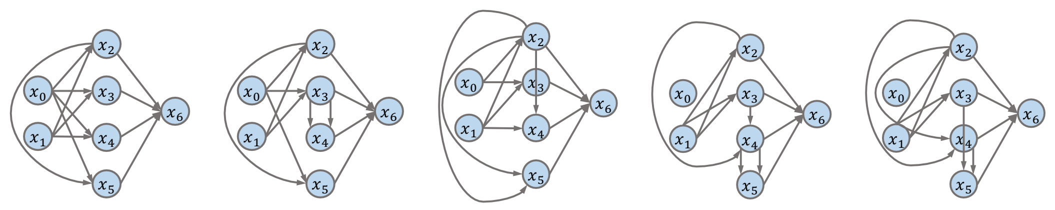
Recently, Xie et al. (2019) have shown that neural networks constructed using cells of random connections can achieve state-of-the-art performance on multiple datasets. Taking this step further, we wonder whether popular NAS cells share any common connection patterns, which may explain why these cells are chosen during the architecture search. To investigate the connection patterns, we sketch the topologies of popular NAS cells by omitting the coupling operations.

Figure 1 illustrates the topologies of 5 popular NAS cells[3]. To examine the connection pattern precisely, we introduce the concept of the depth and the width for a cell. The depth of a cell is defined as the number of connections on the longest path from input nodes to the output node. The width of a cell is defined as the summation of the width of the intermediate nodes that are only connected to the input nodes. The width of a node is the number of channels if it is the output from a convolution operation; if the node is the output of the linear operation, the width of is the dimension of the output features. Supposing the width of each intermediate node is c, as shown in Figure 1, the width and depth of the SNAS (Xie et al., 2018) cell are $4c$ and 2 respectively, and the width and depth of the AmoebaNet (Real et al., 2018) cell are $3c$ and 4 correspondingly.

Following the definitions above, the smallest depth and largest width for a 7-node cell with 2-node inputs are 2 and $4c$ respectively. Similarly, for an 8-node with the same setting, the smallest depth and largest width of a cell are 2 and $5c$ respectively. From Figure 1, we therefore observe that cells from popular NAS architectures, with width close to $4c/5c$ and depth close to 2, tend to be widest and shallowest among all candidate cells from the same search space. Regarding it as the common connection pattern, we achieve the following observation:

[2]There is usually another type of cell with stride 2 for the dimension reduction purpose, called *reduction cell*. We omit it here for brevity.

[3]We only visualize normal cell since the number of normal cells is significantly larger than the reduction cells in NAS architectures.



(a) C^{darts}, $3c$, 2 (b) C_1^{darts}, $2c$, 3 (c) C_2^{darts}, $2c$, 3 (d) C_3^{darts}, $2c$, 4 (e) C_4^{darts}, $2c$, 4

Figure 2: Topologies of DARTS (Liu et al., 2018) cell (leftmost) and its variants with random connections. Each sub-figure also reports the name of a cell as well as its width and depth respectively. The width of a cell is calculated by assuming that all intermediate nodes share the same width c. The depth and width of cells are increasing and decreasing respectively from left to right. Namely, the original DARTS cell C^{darts} is the widest and shallowest one among these cells.

Observation 3.1 (The Common Connection Pattern) *NAS architectures generated by popular NAS algorithms tend to have the widest and shallowest cells among all candidate cells in the same search space.*

4 The Impacts of The Common Connections on Optimization

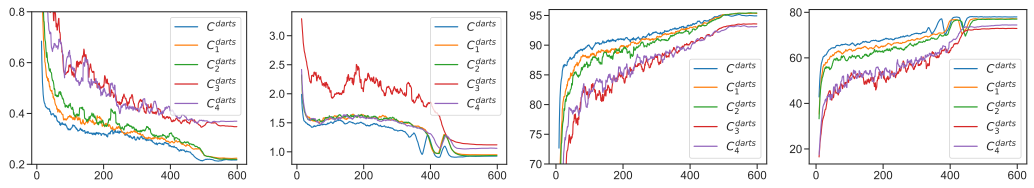
Given that popular NAS cells share the common connection pattern, we then explore the impacts of this common connection pattern from the optimization perspective to answer the question: *why these connections are selected during architecture search?* We sample and train variants of popular NAS architectures with random connections following the sampling method in Appendix A.2 and training details in Appendix A.3. Comparing randomly connected variants to the popular NAS architectures, we find that architectures with the common connection pattern indeed converge faster so as to be chosen by NAS algorithms (in Section 4.1). To understand why the common connection pattern contributes to faster convergence, we further investigate the loss landscape and gradient variance of Popular NAS architectures and their variants via both empirical experiments (in Section 4.2) and theoretical analysis (in Section 4.3).

4.1 Convergence

Popular NAS algorithms conventionally evaluate the performance of a candidate architecture prematurely without waiting for fully converged model parameters during the search. For instance, DARTS (Liu et al., 2018), SNAS (Xie et al., 2018) and ENAS (Pham et al., 2018) optimize hyper-parameters of architectures and model parameters concurrently. The training time of each candidate architecture therefore is short, which is far from the requirement for full convergence. Meanwhile, AmoebaNet (Real et al., 2018) obtains the performance of candidate architectures trained with only a few epochs. In other words, these candidate architectures are not compared based on their generalization performance at convergence. Consequently, architectures with faster convergence rates are more likely to be selected because their validation performance is higher, given the same number of training epochs.

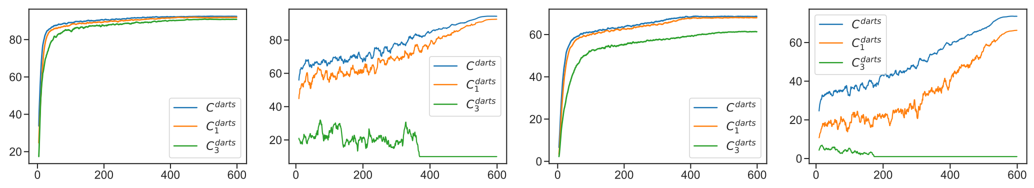
Since popular NAS architectures are selected during architecture search and they share the common connection pattern, we examine the relation between the common connection pattern and their fast convergence in this section. To derive the relation, we compare the convergence of original NAS architectures and their variants with random connections in the cell via empirical studies. We first sample variants of popular NAS cells following the sampling method in Appendix A.2. Then, we train original NAS architectures and their random variants on CIFAR-10 and CIFAR-100 following the training details in Appendix A.3. During training, we evaluate the testing loss and accuracy of these architectures. Since the convergence is also related to the optimization settings, we also evaluate their convergence performance under different learning rate to further evaluate the impacts of the common connection pattern.

Take DARTS (Liu et al., 2018) for example, Figure 2 shows the connection topology of the original DARTS cell and its random variants. Figure 3 shows the test loss and accuracy curves during train-



(a) CIFAR-10 Loss (b) CIFAR-100 Loss (c) CIFAR-10 Accuracy (d) CIFAR-100 Accuracy

Figure 3: Test loss and test accuracy (%) curves of DARTS and its randomly connected variants on CIFAR-10 and CIFAR-100 during training. The default learning rate is 0.025.



(a) CIFAR-10, 0.0025 (b) CIFAR-10, 0.25 (c) CIFAR-100, 0.0025 (d) CIFAR-100, 0.25

Figure 4: Test accuracy (%) curves of DARTS and its randomly connected variants on CIFAR-10 and CIFAR-100 during training under different learning rate (0.0025 and 0.25). We only evaluate C^{darts}, C_1^{darts} and C_3^{darts} for illustration. The caption of each sub-figure reports the dataset and the learning rate.

ing. As illustrated in Figure 3, cell C^{darts}, known as the widest and shallowest cell, has the fastest and most stable convergence compared with its variants. Furthermore, its converged performance, including test loss and test accuracy, is the best among these cells. While C_1^{darts} and C_2^{darts} have slightly larger depth and smaller width than C^{darts} as shown in Figure 2, their convergence and the converged performance are similar to C^{darts} but with more agitated curves. In contrast, C_3^{darts} and C_4^{darts} have the smallest width and largest depth among these 5 cells, and its loss curve is the most unstable one. Besides, its converged performance is significantly worse than that of C^{darts}.

Figure 4 further validates the difference of convergence under different learning rates. The difference in terms of convergence rate and stability is more obvious between C^{darts} and its variants with a larger learning rate as shown in Figure 4. Interestingly, C_3^{darts} completely fail to converge on both CIFAR-10 and CIFAR-100 with a larger learning rate of 0.25. While there is only minor difference among these cells with a smaller learning rate of 0.0025, we still find that there is a decreasing performance of convergence (i.e., convergence rate and stability) from C^{darts}, C_1^{darts} to C_3^{darts}. Overall, the observations are consistent with the results in Figure 3.

The results of other popular NAS architectures are reported in Appendix B.2. Combining with all these observations, we discover that the popular NAS cells with common connection pattern indeed converge faster, which explains why these popular NAS cells are selected during architecture search. Besides, another distinguished property of the popular NAS cells with the common connection pattern is that their optimization is more stable. The next question is then *why the common connection pattern leads to a faster and more stable convergence?*

4.2 Empirical Study of Factors Affecting Convergence

Since the common connection pattern is related to fast convergence, we refer to the theoretical convergence analysis to investigate the cause of fast convergence. In this section, we first introduce the convergence analysis (i.e., Theorem 4.1) of non-convex optimization with the randomized stochastic gradient method (Ghadimi & Lan, 2013). Based on the analysis, we introduce the possible factors related to the common connection pattern that may affect the convergence. We then examine these factors empirically in the following subsections.

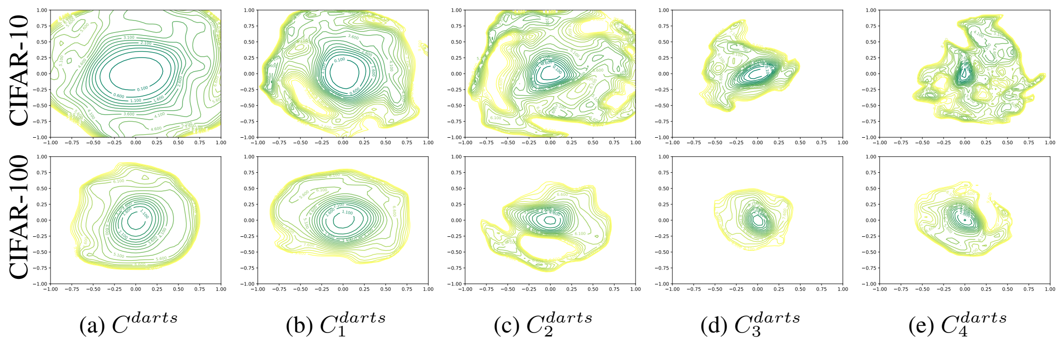


Figure 5: Loss contours of DARTS and its variants with random connections on the test dataset of CIFAR-10. The lighter color of the contour lines indicates a larger loss. Notably, the loss of the blank area, around the corners of each plot, is extremely large. Besides, the area with denser contour lines indicates a steeper loss surface.

Theorem 4.1 *(Ghadimi & Lan, 2013) Let f be L-smooth and non-convex function, and let f^* be the minimal. Given repeated, independent accesses to stochastic gradients with variance bound σ^2 for $f(\boldsymbol{w})$, SGD with initial $\boldsymbol{w}_0$, total iterations $N > 0$ and learning rate $\gamma_k < \frac{1}{L}$ achieves the following convergence by randomly choosing $\boldsymbol{w}_k$ as the final output $\boldsymbol{w}_R$ with probability $\frac{\gamma_k}{H}$ where $H = \sum_{k=1}^{N} \gamma_k$:*

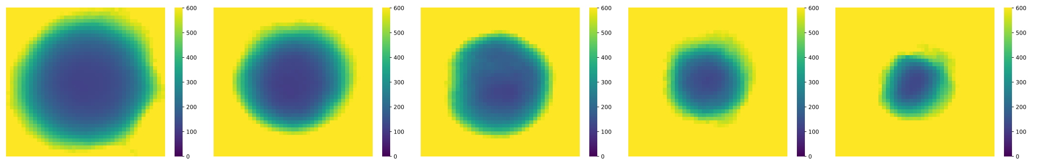
$$\mathbb{E}[\|\nabla f(\boldsymbol{w}_R)\|^2] \leq \frac{2(f(\boldsymbol{w}_0) - f^*)}{H} + \frac{L\sigma^2}{H}\sum_{k=1}^{N}\gamma_k^2$$

In this paper, f and $\boldsymbol{w}$ denote the objective (loss) function and model parameters respectively. Based on the theorem, Lipschitz smoothness L and gradient variance σ^2 significantly affect the convergence, including the rate of convergence as well as its stability. Particularly, a smaller Lipschitz constant L or smaller gradient variance σ^2 would lead to a smaller convergence error and damped oscillations, which indicates a faster and more stable convergence. Since the Lipschitz constant L and gradient variance σ^2 represents the properties of the objective function, they are likely to be related to the NAS architecture. In the following subsections, we therefore conduct both empirical and theoretical analysis for the impacts of the common connection pattern on the Lipschitz smoothness and gradient variance.

4.2.1 Loss Landscape

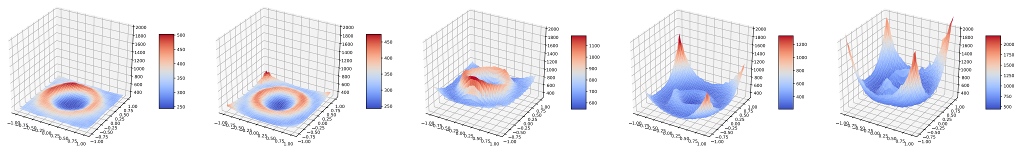
The constant L of Lipschitz smoothness is closely correlated with the Hessian matrix of the objective function shown by Nesterov (2004), which requires substantial computation and can only represent the global smoothness. The loss contour, which has been widely adopted to visualize the loss landscape of neural networks by Goodfellow & Vinyals (2015); Li et al. (2018), are instead computationally efficient and are able to report the local smoothness of the objective function. We therefore extend the method in Li et al. (2018) to plot the loss contour of function $s(\alpha, \beta) = \mathbb{E}_{i\sim P}[f_i(\boldsymbol{w}^* + \alpha\boldsymbol{w}_1 + \beta\boldsymbol{w}_2)]$ to explore the loss landscape of popular NAS architectures. The notation $f_i(\cdot)$ denotes the loss evaluated at i_{th} instance and P denotes the distribution of dataset. $\boldsymbol{w}^*$, $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ denote the (local) optimal and the two direction vectors randomly sampled from Gaussian distribution respectively. And α, β denote the step sizes to perturb $\boldsymbol{w}^*$.

To study the impact of common connection pattern on Lipschitz smoothness, we compare the loss landscape between popular NAS architectures and their randomly connected variants trained in Section 4.1 on CIFAR-10 and CIFAR-100. Due to the space limitation, we only plot loss landscape of DARTS (Liu et al., 2018) and its randomly connected variants in Figure 5. We observe that the connection topology has a significant influence on the smoothness of loss landscape. With the widest and shallowest cell, C^{darts} has a fairly benign and smooth landscape along with the widest near-convex region around the optimal. With a deeper and narrower cell, C_1^{darts} and C_2^{darts} have a more agitated loss landscape compared with C^{darts}. Furthermore, C_3^{darts}, with the smallest width



(a) C^{darts} (b) C_1^{darts} (c) C_2^{darts} (d) C_3^{darts} (e) C_4^{darts}

Figure 6: Heat maps of the gradient variance from DARTS and its randomly connected variants around the optimal on the test dataset of CIFAR-10. The lighter color indicates a larger gradient variance. Notably, the gradient variance of the yellow area, around the corners of each plot, is extremely large. Obviously, the region with relatively small gradient variance becomes smaller from left to right.



(a) C^{darts} (b) C_1^{darts} (c) C_2^{darts} (d) C_3^{darts} (e) C_4^{darts}

Figure 7: 3D surfaces of the gradient variance from DARTS and its randomly connected variants around the optimal on the test dataset of CIFAR-100. The height of the surface indicates the value of gradient variance. Notably, the height of gradient variance surface is gradually increasing from left to right. Especially, C^{darts} has the smoothest and lowest surface of gradient variance among these architectures.

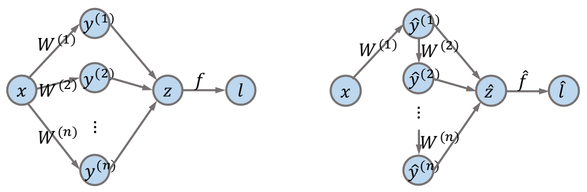
and largest depth among these cells, has the most complicated loss landscape and the narrowest and steepest near-convex region around the optimum. Moreover, its loss becomes extremely large when moving from the center to the outside along any direction with small steps. Since Lipschitz is positively correlated with the largest eigenvalue of Hessian matrix (Nesterov, 2004) indicating the maximum curvature of the objective function, a smaller Lipschitz constant L corresponds to a smoother loss landscape. C^{darts} then achieve the smallest Lipschitz constant among these cells.

Consistent results can be found in Appendix B.3 for the loss landscape of other popular NAS cells and their variants. Based on these results, we conclude that increasing the width and decreasing the depth of a cell widens the near-convex region around the optimal and smooths the loss landscape. The constant L of Lipschitz smoothness therefore becomes smaller locally and globally. Following Theorem 4.1, architectures with the common connection pattern shall converge faster and more stably.

4.2.2 Gradient Variance

The gradient variance indicates the noise level of gradient by randomly selecting training instances in stochastic gradient descent (SGD) method. Large gradient variance indicates large noise in the gradient, which typically results in unstable updating of model parameters. Following Ghadimi & Lan (2013), gradient variance is defined as $\mathrm{Var}(\nabla f_i(\boldsymbol{w}))$ in this paper where $f_i(\cdot)$ denotes the loss evaluated over the i_{th} instance. Similar to the visualization of loss landscape in Section 4.2.1, we visualize the gradient variance by $g(\alpha, \beta) = \mathrm{Var}(\nabla f_i(\boldsymbol{w}^* + \alpha \boldsymbol{w}_1 + \beta \boldsymbol{w}_2))$. All notations have the same meaning as shown in Section 4.2.1. For better visualization, we plot the figures using the standard deviation, i.e., $\sqrt{g(\alpha, \beta)}$, to avoid extremely large values.

To study the impact of the common connection pattern on the gradient variance, we compare the gradient variance between popular NAS architectures and their randomly connected variants trained in Section 4.1 on CIFAR-10 and CIFAR-100. We visualize the gradient variance of DARTS (Liu et al., 2018) and its randomly connected variants in Figure 6 and Figure 7. Similarly, we have observed the relation between the common connection pattern and the gradient variance around the optimal. Obviously, with the width of a cell decreasing and the depth increasing (i.e., from



(a) case I: widest cell (b) case II: narrowest cell

Figure 8: Two architectures to compare in the theoretical analysis: (a) architecture with widest cell; (b) architecture with narrowest cell. The notation l and $\widehat{l}$ denote the values of objective function f and $\widehat{f}$ evaluated at input $\boldsymbol{x}$ respectively.

C^{darts} to C_4^{darts}), the region with relatively small gradient variance becomes smaller as shown in Figure 6. Moreover, the gradient variance generally shows an increasing trend from C^{darts} to C_4^{darts} in Figure 7. Consequently, the gradient becomes noisier in the neighborhood of the optimal, which typically make the optimization harder and unstable. The training of C^{darts} therefore shall be the fastest and most stable one among these random cells based on Theorem 4.1, which is consistent with the convergence shown in Figure 3 and Figure 4.

Similar results from other popular cells are provided in Appendix B.4. Based on these results, we note that the increase of width and the decreasing of the depth of a cell result in a smaller gradient variance, which makes the optimization process less noisy and more efficient. The convergence therefore shall be fast and stable following Theorem 4.1.

4.3 Theoretical Analysis of Factors Affecting Convergence

Our empirical study so far suggests that the common connection pattern smooths the loss landscape and decreases the gradient variance. Popular NAS architectures therefore converge fast. In this section, we explain the impacts of common connection pattern on Lipschitz smoothness and gradient variance from a theoretical perspective. All proofs can be found in Appendix C.

4.3.1 Setup

We analyze the impact of the common connection pattern by comparing the Lipschitz smoothness and gradient variance between the architecture with the widest cell and the architecture with the narrowest cell as shown in Figure 8. To simplify the analysis, the cells we investigate contain only one input node $\boldsymbol{x}$ and one output node represented as vectors. The input node can be training instances or output node from any proceeding cell. Moreover, all operations in the cell are linear operations without any non-linear activation function. Suppose there are n intermediate nodes in a cell, the i_{th} intermediate node and its associated weight matrix are denoted as $\boldsymbol{y}^{(i)}$ and $W^{(i)}(i = 1, \cdots, n)$ respectively. The output node $\boldsymbol{z}$ denotes the concatenation of all intermediate nodes. Both cells have the same arbitrary objective function f following the output node, which shall consist of the arbitrary number of activation functions and cells. For clarity, we refer to the objective function, intermediate nodes and output node of the architecture with the narrowest cell as $\widehat{f}$, $\widehat{\boldsymbol{y}}^{(i)}$ and $\widehat{\boldsymbol{z}}$ respectively. As shown in Figure 8, the intermediate node $\boldsymbol{y}^{(i)}$ and $\widehat{\boldsymbol{y}}^{(i)}$ can be computed by $\boldsymbol{y}^{(i)} = W^{(i)}\boldsymbol{x}$ and $\widehat{\boldsymbol{y}}^{(i)} = \prod_{k=1}^{i} W^{(k)}\boldsymbol{x}$ respectively. Particularly, we set $\prod_{k=1}^{i} W^{(k)} = W^{(i)}W^{(i-1)} \cdots W^{(1)}$ in this paper.

4.3.2 Theoretical Results

Due to the complexity of the standard Lipschitz smoothness, we instead investigate the block-wise Lipschitz smoothness (Beck & Tetruashvili, 2013) of the two cases shown in Figure 8. In Theorem 4.2, we show that the block-wise Lipschitz constant of the narrowest cell is scaled by the largest eigenvalues of the model parameters (i.e., $W^{(i)}(i = 1, \cdots, n)$). Notably, the Lipschitz constant of the narrowest cell can be significantly larger than the one of the widest cell while most of the largest eigenvalues are larger than 1, which slows down the convergence substantially. The empirical study in Section 4.2.1 has validated the results.

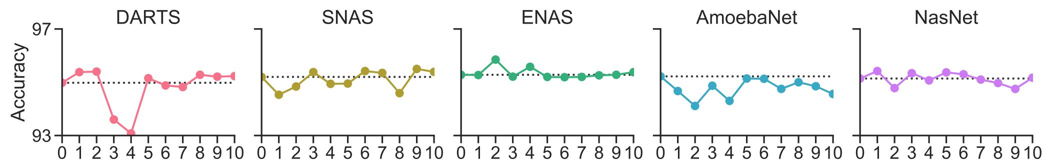


Figure 9: Comparison of the test accuracy at convergence between popular NAS architectures and their randomly connected variants on CIFAR-10. Each popular NAS architecture (index 0 on the x-axis) is followed by 10 randomly connected variants (from index 1 to index 10 on the x-axis). The dashed lines report the accuracy of the popular NAS architectures. Notably, some random variants achieve higher accuracy than the dashed line in each plot except for the plot of AmoebaNet.

Theorem 4.2 (The impact of common connection pattern on block-wise Lipschitz smoothness) *Let $\lambda^{(i)}$ be the largest eigenvalue of $W^{(i)}$. Given the widest cell with objective function f and the narrowest cell with objective function $\widehat{f}$, by assuming the block-wise Lipschitz smoothness of the widest cell as $\left\|\frac{\partial f}{\partial W_1^{(i)}} - \frac{\partial f}{\partial W_2^{(i)}}\right\| \leq L^{(i)} \left\|W_1^{(i)} - W_2^{(i)}\right\|$ for any $W_1^{(i)}$ and $W_2^{(i)}$, the block-wise Lipschitz smoothness of the narrowest cell then can be represented as*

$$\left\|\frac{\partial \widehat{f}}{\partial W_1^{(i)}} - \frac{\partial \widehat{f}}{\partial W_2^{(i)}}\right\| \leq (\prod_{j=1}^{i-1} \lambda^{(j)}) L^{(i)} \left\|W_1^{(i)} - W_2^{(i)}\right\|$$

We then compare the gradient variance of the two cases shown in Figure 8. Interestingly, gradient variance suggests a similar but more significant difference between the two cases compared with their difference in Lipschitz smoothness. As shown in Theorem 4.3, the gradient variance of the narrowest cell is not only scaled by the square of the largest eigenvalue of the weight matrix but also scaled by the number of intermediate nodes. Moreover, the upper bound of its gradient variance has numbers of additional terms, leading to a significantly larger gradient variance. The empirical study in Section 4.2.2 has confirmed the results.

Theorem 4.3 (The impact of common connection pattern on gradient variance) *Let $\lambda^{(i)}$ be the largest eigenvalue of $W^{(i)}$. Given the widest cell with objective function f and the narrowest cell with objective function $\widehat{f}$, by assuming the gradient variance of the widest cell as $\mathbb{E}\left\|\frac{\partial f}{\partial W^{(i)}} - \mathbb{E}\frac{\partial f}{\partial W^{(i)}}\right\|^2 \leq (\sigma^{(i)})^2$ for any $W^{(i)}$, the gradient variance of the narrowest cell is then bounded by*

$$\mathbb{E}\left\|\frac{\partial \widehat{f}}{\partial W^{(i)}} - \mathbb{E}\frac{\partial \widehat{f}}{\partial W^{(i)}}\right\|^2 \leq n \sum_{k=i}^{n} (\frac{\sigma^{(k)}}{\lambda^{(i)}} \prod_{j=1}^{k} \lambda^{(j)})^2$$

5 Generalization beyond the Common Connections

Our empirical and theoretical results so far have demonstrated that the common connection pattern helps to smooth loss landscape and make gradient more accurate. Popular NAS architectures with the common connection pattern therefore converge faster, which explains why popular NAS architectures are selected by the NAS algorithms. However, we have ignored the generalization performance obtained by popular NAS architectures and their random variants. We therefore wonder *whether popular NAS architectures with common connection pattern generalize better*.

In Figure 9, we visualize the test accuracy of popular NAS architectures and their randomly connected variants trained in Section 4.1. Obviously, the popular NAS architectures always achieve competitive accuracy compared with most of the random variants. However, there are some random variants, which achieve higher accuracy than the popular architectures except for C^{amoeba} as shown in Figure 9. Popular NAS architectures with the common connection pattern therefore are not guaranteed to generalize better, although they typically converge faster than other random variants.

Table 1: Comparison of the test error at convergence between the original and the adapted NAS architectures on CIFAR-10 and CIFAR-100. The entire networks are constructed and trained following the experimental settings reported in Appendix A.3, which may slightly deviate from the original ones.

Architecture	**CIFAR-10** (%)		**CIFAR-100** (%)	
	original	**adapted**	**original**	**adapted**
NASNet (Zoph et al., 2018)	**2.65**	2.80	17.06	**16.86**
AmoebaNet (Real et al., 2018)	**2.76**	2.91	17.55	**17.28**
ENAS (Pham et al., 2018)	**2.64**	2.76	16.67	**16.04**
DARTS (Liu et al., 2018)	**2.67**	2.73	16.41	**16.15**
SNAS (Xie et al., 2018)	2.88	**2.69**	17.78	**17.20**

To further examine the impact of common connection pattern on generalization performance, we adapt the connections of popular NAS architectures to obtain the widest and shallowest cells in their original search space. The adaption is possible due to the fact that the cells (including normal cell and reduction cell) of popular NAS architectures are generally not widest and narrowest as shown in Figure 1. While there are various widest and shallowest cells, we follow the connection pattern of SNAS cell shown in Figure 1(e) to obtain the widest and shallowest cells. Notably, the parameter size of the original and adapted architectures are comparable. We then train these architectures with regularization mechanisms in Appendix A.4. Other training details are presented in Appendix A.3.

Table 1 illustrates the comparison of the test accuracy between our adapted NAS architectures and the original ones. As shown in Table 1, the adapted architectures achieve slightly smaller test error on CIFAR-100. Nevertheless, most of the adapted architectures except for the adapted SNAS, obtain larger test error than the original NAS architectures on CIFAR-10. The results again suggest that the common connection pattern may not help architectures generalize better, while they typically achieve competitive generalization performance.

The results above have shown that architectures with the common connection pattern may not generalize better despite of a faster convergence. The results are consistent with the observation that popular NAS architectures fail to achieve better generalization performance compared with the architectures selected by random search, as reported by Li & Talwalkar (2019) and Sciuto et al. (2019). To improve current NAS algorithms, we therefore need to re-think the evaluation of the performance of candidate architectures during architecture search since the current NAS algorithms are not based on the generalization performance at convergence as mentioned in Section 4.1. Nonetheless, architectures with the common connection pattern usually guarantee a stable and fast convergence along with competitive generalization performance, which shall be good prior knowledge for designing architectures and NAS algorithms.

6 Conclusion and Discussion

In this paper, we examine the common connection pattern of NAS architectures. We also investigate the impact of the common connection pattern on the optimization from the loss landscape and gradient variance perspective with both empirical and theoretical study. We explain the reason that the popular NAS architectures are selected during architecture search. Finally, we observe that architectures with the common connection pattern may not generalize better than the candidate architectures.

Since popular NAS architectures consistently achieve fast and stable convergence along with competitive generalization performance, we note that the common connection pattern is good prior knowledge for the design of architectures and NAS algorithms. While architectures generated by existing NAS algorithms fail to achieve the best generalization performance, one promising direction is to refine the performance evaluation during architecture search to make the estimated generalization performance of the candidate architectures more accurate.

REFERENCES

Youhei Akimoto, Shinichi Shirakawa, Nozomu Yoshinari, Kento Uchida, Shota Saito, and Kouhei Nishida. Adaptive stochastic natural gradient method for one-shot neural architecture search. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pp. 171–180. PMLR, 2019.

Amir Beck and Luba Tetruashvili. On the convergence of block coordinate descent type methods. *SIAM Journal on Optimization*, 23(4):2037–2060, 2013.

Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. In *ICLR (Poster)*. OpenReview.net, 2019.

Terrance Devries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout. *CoRR*, abs/1708.04552, 2017.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019.

Saeed Ghadimi and Guanghui Lan. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

Ian J. Goodfellow and Oriol Vinyals. Qualitatively characterizing neural network optimization problems. In *ICLR*, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *CVPR*, pp. 2261–2269. IEEE Computer Society, 2017.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pp. 1106–1114, 2012.

Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. In *ICLR (Poster)*. OpenReview.net, 2017.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *NeurIPS*, pp. 6391–6401, 2018.

Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. *CoRR*, abs/1902.07638, 2019.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu. Neural architecture optimization. In *NeurIPS*, pp. 7827–7838, 2018.

Niv Nayman, Asaf Noy, Tal Ridnik, Itamar Friedman, Rong Jin, and Lihi Zelnik-Manor. XNAS: neural architecture search with expert advice. *CoRR*, abs/1906.08031, 2019.

Yurii Nesterov. *Introductory Lectures on Convex Optimization - A Basic Course*, volume 87 of *Applied Optimization*. Springer, 2004.

Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*, 2018.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. *arXiv preprint arXiv:1802.01548*, 2018.

Christian Sciuto, Kaicheng Yu, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. Evaluating the search phase of neural architecture search. *arXiv preprint arXiv:1902.08142*, 2019.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, pp. 1–9. IEEE Computer Society, 2015.

Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, and Quoc V. Le. Mnasnet: Platform-aware neural architecture search for mobile. *CoRR*, abs/1807.11626, 2018.

Saining Xie, Alexander Kirillov, Ross Girshick, and Kaiming He. Exploring randomly wired neural networks for image recognition. *arXiv preprint arXiv:1904.01569*, 2019.

Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926*, 2018.

Zirui Zhou, Qi Zhang, and Anthony Man-Cho So. 1,p-norm regularization: Error bounds and convergence rate analysis of first-order methods. In *ICML*, pp. 1501–1510, 2015.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. In *CVPR*, pp. 8697–8710. IEEE Computer Society, 2018.

APPENDIX A EXPERIMENTAL SETUP

A.1 DATA PRE-PROCESSING AND AUGMENTATION

Our experiments are conducted on CIFAR-10 and CIFAR-100 (Krizhevsky et al., 2009), which contains 50,000 training images and 10,000 test images of 32×32 pixels in 10 and 100 classes respectively. We adopt exactly the same data pre-processing and argumentation as described in DARTS (Liu et al., 2018): zero padding the training images with 4 pixels on each side and then randomly cropping them back to 32×32 images; randomly flipping training images horizontally; normalizing training images with the means and standard deviations along the channel dimension.

A.2 SAMPLING OF RANDOM CONNECTED VARIANTS

For a N-node NAS cell, there are $\frac{(N-2)!}{(M-1)!}$ possible connections with M input nodes and one output node. There are therefore hundreds to thousands of possible randomly connected variants for each popular NAS cell. Due to the prohibitive cost of comparing popular NAS cell with all variants, we randomly sample some variants to understand why the popular NAS cells are selected.

Given a NAS cell $\boldsymbol{C}$, we fix the partial order of intermediate nodes and their accompanying operations. We then replace the source node of their associated operations by uniformly randomly sampling a node from their proceeding nodes in the same cell. Figure 2 has shown some random connected cells derived from DARTS (Liu et al., 2018). Randomly connected variants of other popular NAS cells are shown in Appendix B.1.

A.3 ARCHITECTURES AND TRAINING DETAILS

For experiments on both CIFAR-10 and CIFAR-100, the neural network architectures are constructed by stacking $L = 20$ cells. Feature maps are down-sampled at the $L/3$-th and $2L/3$-th cell of the entire architecture with stride 2. A more detailed building scheme can be found in DARTS (Liu et al., 2018).

In the default training setting, we apply stochastic gradient descent (SGD) with learning rate 0.025, momentum 0.9, weight decay 3×10^{-4} and batch size 80 to train the models for 600 epochs to ensure the convergence. Learning rate is gradually annealed to zero following the standard cosine annealing schedule. To compare the convergence under different learning rate in Section 4.1, we change the initial learning rate from 0.025 to 0.25 and 0.0025 respectively.

A.4 REGULARIZATION

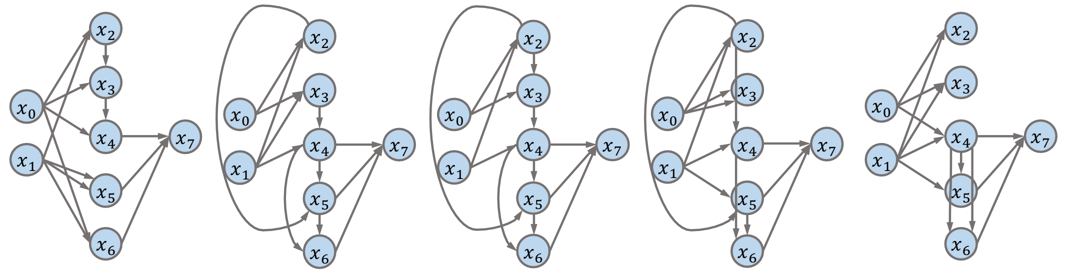
Since regularization mechanisms shall affect the convergence (Zhou et al., 2015), architectures are trained without regularization for a neat empirical study in Section 4. The regularization mechanisms are only used in Section 5 to get the converged generalization performance of the original and adaped NAS architectures on CIFAR-10 and CIFAR-100, shown in Table 1.

There are three adopted regularization mechanisms on both CIFAR-10 and CIFAR-100 in this paper: cutout (Devries & Taylor, 2017), auxiliary tower (Szegedy et al., 2015) and drop path (Larsson et al., 2017). We apply standard cutout regularization with cutout length 16. And the auxiliary tower is located at $2L/3$-th cell of the entire architecture with weight 0.4. We apply the same linearly-increased drop path schedule as in NASNet (Zoph et al., 2018) with the maximum probability of 0.2.

APPENDIX B MORE RESULTS

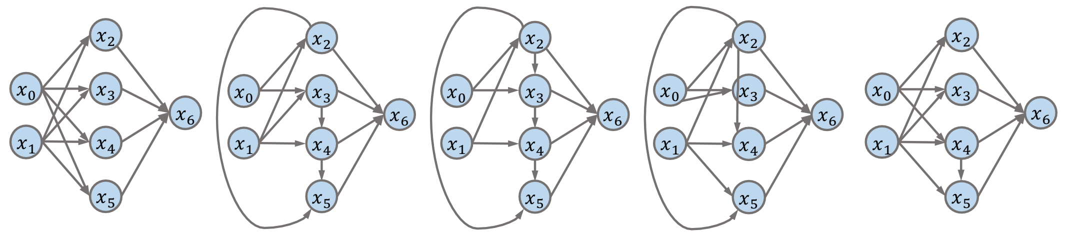
B.1 CONNECTIONS OF NAS ARCHITECTURES AND THEIR VARIANTS

This section compares the connections of the AmoebaNet cell (Real et al., 2018), SNAS cell (Xie et al., 2018) and their variants of random connections, as shown in Figure 10 and 11. The random variants are sampled following the method in Section A.2. We also compute the depth and width of these cells to verify the common connection pattern presented in Section 3.



(a) $C^{amoebanet}$,3c,4 (b) $C_1^{amoebanet}$,2c,5 (c) $C_2^{amoebanet}$,c,6 (d) $C_3^{amoebanet}$,2c,4 (e) $C_4^{amoebanet}$,3c,3

Figure 10: Connection topology of AmoebaNet cell (Real et al., 2018) and its randomly connected variants. Each sub-figure reports the name of a cell as well as its width and depth respectively. The leftmost one is the original connection from AmoebaNet normal cell and others are the ones randomly sampled. The width of a cell is also calculated by assuming that each intermediate node shares the same width c. Notably, the original AmoebaNet cell has the largest width and almost the smallest depth among these cells.

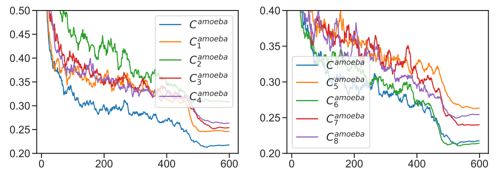


(a) C^{snas}, 4c, 2 (b) C_1^{snas}, 2c, 4 (c) C_2^{snas}, c, 5 (d) C_3^{snas}, 2c, 3 (e) C_4^{snas}, 3c, 3

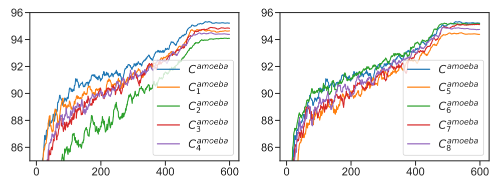
Figure 11: Connection topology of SNAS cell with mild constraint (Xie et al., 2018) and its randomly connected variants. Each sub-figure reports the name of a cell as well as its width and depth respectively. The leftmost one is the original connection from SNAS normal cell and others are the ones randomly sampled. The width of a cell is also calculated by assuming that each intermediate node shares the same width c. Notably, the original SNAS cell has the largest width and the smallest depth among these cells.

B.2 Convergence

In this section, we plot more test loss and accuracy curves on CIFAR-10 (Krizhevsky et al., 2009) for original popular NAS architectures and their (8) randomly connected variants, as shown in Figure 12 and Figure 13. Consistent with Section 4.1, the popular NAS cells, with larger width and smaller depth, typically achieve faster and more stable convergence than the random variants.

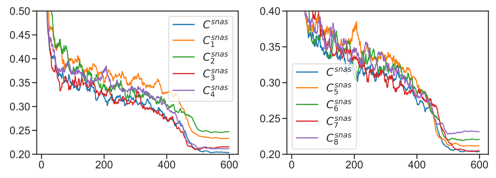


(a) Test loss

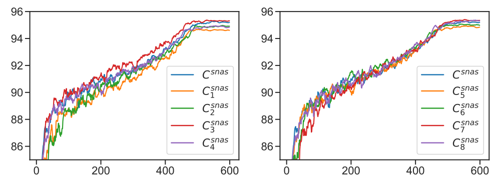


(b) Test acc

Figure 12: Test loss and test accuracy curves of AmoebaNet (Real et al., 2018) and its variants on CIFAR-10 during training. We observe that while AmoebaNet has the widest and shallowest cell, its convergence is the fastest and most stable one. The curves of other cells are more agitated.

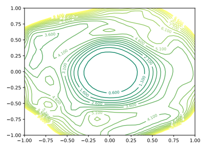


(a) Test loss

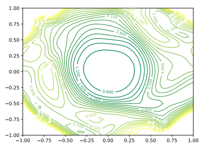


(b) Test accuracy

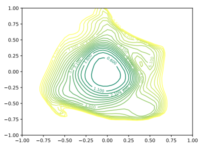
Figure 13: Test loss and test accuracy curves of SNAS (Xie et al., 2018) and its variants on CIFAR-10 during training. Although these 5 loss curves are very close, we can still distinguish that the SNAS cell is one of the cells with almost the fastest and most stable convergence.



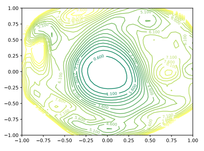
(a) $\boldsymbol{C}^{amoeba}$



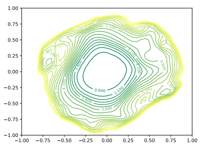
(b) $\boldsymbol{C}_1^{amoeba}$



(c) $\boldsymbol{C}_2^{amoeba}$



(d) $\boldsymbol{C}_3^{amoeba}$



(e) $\boldsymbol{C}_4^{amoeba}$

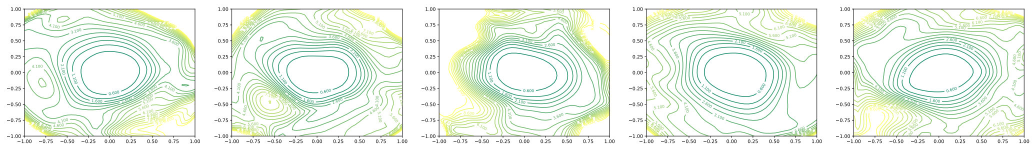
Figure 14: Loss landscape of AmoebaNet (Real et al., 2018) and its variants of random connections on the test dataset of CIFAR-10. With the widest and shallowest cell, the landscape of $\boldsymbol{C}^{amoebanet}$ is smoothest and benignest among these 5 architectures. In contrast, the width and the depth of $\boldsymbol{C}_2^{amoebanet}$ are the smallest and largest respectively; it has a narrowest and steepest near-convex region surrounding the optimal. Although $\boldsymbol{C}_4^{amoebanet}$ shows the exceptional loss landscape, the overall pattern is that a wider and shallower cell has a smoother and benigner optimization landscape.

B.3 Loss Landscape

In this section, we visualize loss landscapes for more popular NAS architectures and their randomly connected variants, as shown in Figure 14 and Figure 15. Notably, with wider and shallower cell, the popular NAS architectures have a smoother and benigner loss landscape, which further supports the results in Section 4.2.1.

B.4 Gradient Variance

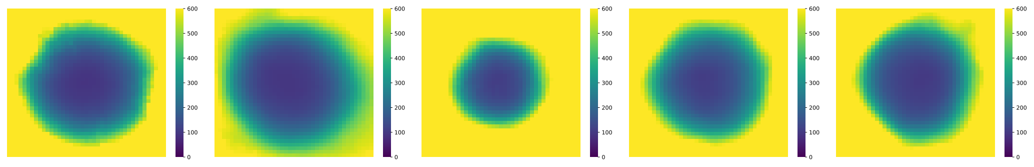
In this section, we visualize the gradient variance for more popular NAS architectures as well as their variants with random connection, such as AmoebaNet in Figure 16, SNAS in Figure 17. Notably,



(a) $\boldsymbol{C}^{snas}$ (b) $\boldsymbol{C}_1^{snas}$ (c) $\boldsymbol{C}_2^{snas}$ (d) $\boldsymbol{C}_3^{snas}$ (e) $\boldsymbol{C}_4^{snas}$

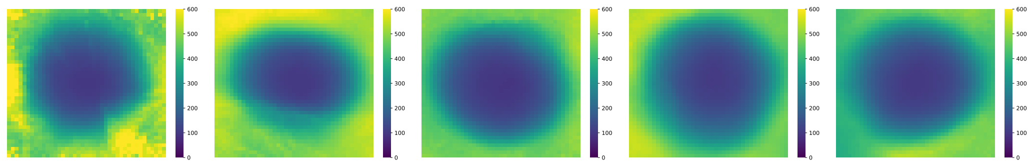
Figure 15: Loss landscape of SNAS (Xie et al., 2018) and its variants of random connections on the test dataset of CIFAR-10. With the largest depth and smallest width, $\boldsymbol{C}_2^{snas}$ has the narrowest and steepest optimization landscape. Interestingly, $\boldsymbol{C}_3^{snas}$, with the widest and smoothest optimization landscape, has a larger depth and smaller width than $\boldsymbol{C}^{snas}$. However, the difference between $\boldsymbol{C}^{snas}$ and $\boldsymbol{C}_3^{snas}$ is small. Overall, we can also conclude that a wide and shallow cell has a benign optimization landscape.

with wider and shallower cell, the popular NAS architectures achieve relatively smaller gradient variance, which further confirms the results in Section 4.2.2.



(a) $\boldsymbol{C}^{amoeba}$ (b) $\boldsymbol{C}_1^{amoeba}$ (c) $\boldsymbol{C}_2^{amoeba}$ (d) $\boldsymbol{C}_3^{amoeba}$ (e) $\boldsymbol{C}_4^{amoeba}$

Figure 16: Heat maps of gradient variance from AmoebaNet (Real et al., 2018) and its variants of random connections on the test dataset of CIFAR-10. Although original AmoebaNet does not achieve the smallest gradient variance around optimal, its gradient variance is still competitive to the other cells, especially for $C_2^{amoebanet}$. While all other cells have larger width and smaller depth than $C_2^{amoebanet}$, their gradient variance is much smaller.



(a) $\boldsymbol{C}^{snas}$ (b) $\boldsymbol{C}_1^{snas}$ (c) $\boldsymbol{C}_2^{snas}$ (d) $\boldsymbol{C}_3^{snas}$ (e) $\boldsymbol{C}_4^{snas}$

Figure 17: Heat maps of gradient variance from SNAS (Xie et al., 2018) and its variants of random connections on the test dataset of CIFAR-10. Interestingly, the gradient variance of these cells is almost the same, which is small in general. It is therefore hard to differentiate the best one. After checking the computation of SNAS cell, we find that there are many skip-connections chosen as the operations in the cell, which are also helpful to widen cells. Therefore, these cells share small gradient variance.

APPENDIX C THEORETICAL ANALYSIS

C.1 BASICS

We firstly compare the gradient of case I and case II shown in Figure 8. For case I, since $\boldsymbol{y}^{(i)} = W^{(i)}\boldsymbol{x}$, the gradient to each weight matrix $W^{(i)}$ is denoted by

$$\frac{\partial f}{\partial W^{(i)}} = \frac{\partial f}{\partial \boldsymbol{y}^{(i)}}\boldsymbol{x}^T \tag{1}$$

Similarly, since $\widehat{\boldsymbol{y}}^{(i)} = \prod_{k=1}^{i} W^{(k)}\boldsymbol{x}$ for the case II, the gradient to each weight matrix $W^{(i)}$ is denoted by

$$\begin{aligned}
\frac{\partial \widehat{f}}{\partial W^{(i)}} &= \sum_{k=i}^{n}(\prod_{j=i+1}^{k} W^{(j)})^T \frac{\partial \widehat{f}}{\partial \widehat{\boldsymbol{y}}^{(k)}}(\prod_{j=1}^{i-1} W^{(j)}\boldsymbol{x})^T &(2)\\
&= \sum_{k=i}^{n}(\prod_{j=i+1}^{k} W^{(j)})^T \frac{\partial \widehat{f}}{\partial \widehat{\boldsymbol{y}}^{(k)}}\boldsymbol{x}^T(\prod_{j=1}^{i-1} W^{(j)})^T &(3)\\
&= \sum_{k=i}^{n}(\prod_{j=i+1}^{k} W^{(j)})^T \frac{\partial f}{\partial W^{(k)}}(\prod_{j=1}^{i-1} W^{(j)})^T &(4)
\end{aligned}$$

Exploring the fact that $\frac{\partial \widehat{f}}{\partial \widehat{\boldsymbol{y}}^{(i)}} = \frac{\partial f}{\partial \boldsymbol{y}^{(i)}}$, we get (4) by inserting (1) into (3).

C.2 PROOF OF THEOREM 4.2

Due to the complexity of comparing the standard Lipschitz constant of the smoothness for these two cases, we instead investigate the block-wise Lipschitz constant (Beck & Tetruashvili, 2013). In other words, we evaluate the Lipschitz constant for each weight matrix $W^{(i)}$ while fixing all other matrices. Formally, we assume the block-wise Lipschitz smoothness of case I as

$$\left\|\frac{\partial f}{\partial W_1^{(i)}} - \frac{\partial f}{\partial W_2^{(i)}}\right\| \leq L^{(i)}\left\|W_1^{(i)} - W_2^{(i)}\right\| \quad \forall\ W_1^{(i)}, W_2^{(i)} \tag{5}$$

The default matrix norm we adopted is 2-norm. And $W_1^{(i)}, W_2^{(i)}$ denote possible assignments for $W^{(i)}$.

Denoting that $\lambda^{(i)} = \left\|W^{(i)}\right\|$, which is the largest eigenvalue of matrix $W^{(i)}$, we can get the smoothness of case II as

$$\begin{aligned}
\left\|\frac{\partial \widehat{f}}{\partial W_1^{(i)}} - \frac{\partial \widehat{f}}{\partial W_2^{(i)}}\right\| &= \left\|\sum_{k=i}^{n}(\prod_{j=i+1}^{k} W^{(j)})^T(\frac{\partial f}{\partial W_1^{(k)}} - \frac{\partial f}{\partial W_2^{(k)}})(\prod_{j=1}^{i-1} W^{(j)})^T\right\| &(6)\\
&\leq \sum_{k=i}^{n}\left\|(\prod_{j=i+1}^{k} W^{(j)})^T(\frac{\partial f}{\partial W_1^{(k)}} - \frac{\partial f}{\partial W_2^{(k)}})(\prod_{j=1}^{i-1} W^{(j)})^T\right\| &(7)\\
&\leq \sum_{k=i}^{n}(\frac{1}{\lambda^{(i)}}\prod_{j=1}^{k}\lambda^{(j)})L^{(k)}\left\|W_1^{(k)} - W_2^{(k)}\right\| &(8)\\
&\leq (\prod_{j=1}^{i-1}\lambda^{(j)})L^{(i)}\left\|W_1^{(i)} - W_2^{(i)}\right\| &(9)
\end{aligned}$$

We get the equality in (6) since $j > i$ and $W^{(j)}$ keeps the same for the computation of block-wise Lipschitz constant of $W^{(i)}$. Based on the triangle inequality of norm, we get (7) from (6). We get (8) from (7) based on the inequality $\|WV\| \le \|W\| \, \|V\|$ and the assumption of the smoothness for case I in (5). Finally, since we are evaluating the block-wise Lipschitz constant for $W^{(i)}$, $W_1^{(k)} = W_2^{(k)}$ while $k \neq i$, which leads to the final inequality (9).

C.3 Proof of Theorem 4.3

Similarly, we assume the gradient variance of case I is bounded as

$$\mathbb{E}\left\|\frac{\partial f}{\partial W^{(i)}} - \mathbb{E}\frac{\partial f}{\partial W^{(i)}}\right\|^2 \le (\sigma^{(i)})^2 \tag{10}$$

The gradient variance of case II is then bounded by

$$\begin{aligned}
\mathbb{E}\left\|\frac{\partial \widehat{f}}{\partial W^{(i)}} - \mathbb{E}\frac{\partial \widehat{f}}{\partial W^{(i)}}\right\|^2 &= \mathbb{E}\left\|\sum_{k=i}^{n}(\prod_{j=i+1}^{k} W^{(j)})^T(\frac{\partial f}{\partial W^{(k)}} - \mathbb{E}\frac{\partial f}{\partial W^{(k)}})(\prod_{j=1}^{i-1} W^{(j)})^T\right\|^2 & (11)\\
&\le n\mathbb{E}\sum_{k=i}^{n}\left\|(\prod_{j=i+1}^{k} W^{(j)})^T(\frac{\partial f}{\partial W^{(k)}} - \mathbb{E}\frac{\partial f}{\partial W^{(k)}})(\prod_{j=1}^{i-1} W^{(j)})^T\right\|^2 & (12)\\
&\le n\sum_{k=i}^{n}(\frac{\sigma^{(k)}}{\lambda^{(i)}}\prod_{j=1}^{k}\lambda^{(j)})^2 & (13)
\end{aligned}$$

We get (12) from (11) based on Cauchy-Schwarz inequality. Based on the inequality $\|WV\| \le \|W\| \, \|V\|$ and the assumption of bounded gradient variance for case I in (10), we get the final inequality.