Unchain the Search Space
with Hierarchical Differentiable Architecture Search

Guanting Liu, Yujie Zhong, Sheng Guo, Matthew R. Scott, Weilin Huang[*]

Malong LLC

{gualiu, jaszhong, sheng, mscott, whuang}@malongtech.com

Abstract

Differentiable architecture search (DAS) has made great progress in searching for high-performance architectures with reduced computational cost. However, DAS-based methods mainly focus on searching for a repeatable cell structure, which is then stacked sequentially in multiple stages to form the networks. This configuration significantly reduces the search space, and ignores the importance of connections between the cells. To overcome this limitation, in this paper, we propose a Hierarchical Differentiable Architecture Search (H-DAS) that performs architecture search both at the cell level and at the stage level. Specifically, the cell-level search space is relaxed so that the networks can learn stage-specific cell structures. For the stage-level search, we systematically study the architectures of stages, including the number of cells in each stage and the connections between the cells. Based on insightful observations, we design several search rules and losses, and mange to search for better stage-level architectures. Such hierarchical search space greatly improves the performance of the networks without introducing expensive search cost. Extensive experiments on CIFAR10 and ImageNet demonstrate the effectiveness of the proposed H-DAS. Moreover, the searched stage-level architectures can be combined with the cell structures searched by existing DAS methods to further boost the performance. Code is available at: https://github.com/msight-tech/research-HDAS
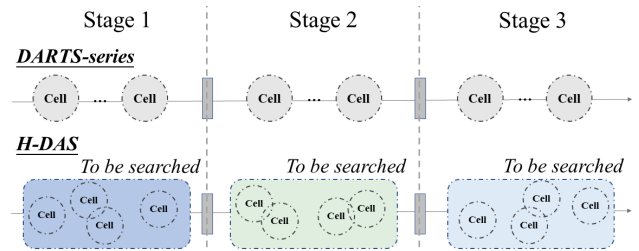
Figure 1: DARTS-series methods search for a repeatable cell structure, which is stacked sequentially in multiple stages. The proposed H-DAS enables a *stage-specific* search of both cell-level and stage-level structures, allowing for a diversity of cell structures, cell distribution and cell connections over different stages. This results in a significant larger search space with more meaningful architecture search.

Introduction

A large number of neural networks or architectures has been designed for various computer vision tasks in the past years (Krizhevsky, Sutskever, and Hinton 2012; Simonyan and Zisserman 2014; Szegedy et al. 2015; He et al. 2016; Huang et al. 2017), where human experts played an important role. Such manually-designed architectures have been proved to be effective, but heavily depend on human skills and experience. Recently, Neural Architecture Search (NAS) has attracted increasing attentions (Baker et al. 2016; Zoph and Le 2016; Pham et al. 2018), and achieved the state-of-the-art performance in various computer vision tasks.

Reinforcement learning (Zoph and Le 2016; Pham et al. 2018) or evolutionary algorithm (Real et al. 2019, 2017) has

been introduced to NAS, due to the discrete nature of the architecture space. However, these methods usually require up to thousands of GPU days (Zoph and Le 2016). A few methods have been developed to reduce the computational cost, such as (Cai, Zhu, and Han 2018; Xie et al. 2018; Dong and Yang 2019; You et al. 2020; Zhong et al. 2020). Among them, differentiable architecture search (Liu, Simonyan, and Yang 2018) attempted to approximate the discrete search space into a continuous one, where gradient descent can be used to optimize the architectures and model parameters jointly. This line of search approaches, referred as DARTS-series methods (Chen et al. 2019; Xu et al. 2019; Chen and Hsieh 2020), has made significant improvements in search speed, while maintaining comparable performance.

However, these DARTS-series methods have two major limitations in terms of search space. First, they commonly perform a cell-level search and adopt the same searched cell structure repeatedly for multiple stages (separated by the reduction cells), which may make the cell structure sub-optimal in the stages, since the optimal cell structures (including the connections and kernel sizes) at different stages can be significantly different. For example, in the architectures searched by (Zoph and Le 2016; Cai, Zhu, and Han 2018), the operations in shallow layers are mainly 3×3 convolutions, while many larger kernels appear in the deeper layers. Second, previous DARTS-series methods mainly fo-

cus on searching for a repeatable cell structure, which is stacked sequentially to form the networks with three stages. This configuration assumes a simple chain-like structure at the stage level, which reduces the search space considerably, and ignores the importance of stage-level connections or structures. As revealed by (Yang, Esperança, and Carlucci 2019), the overall stage-level connections can impact to the final performance of the networks considerably.

In this work, we redesign the search space of DARTS-series methods, and propose a Hierarchical Differentiable Architecture Search (H-DAS) that enables the search of both cell-level (micro-architecture) and stage-level structures (macro-architecture) (Figure 1). H-DAS significantly increases the search space comparing to previous methods. Specifically, for the micro-architecture, we relax the cell-level search space so that the networks can learn the optimized cell structures at different stages. For searching the macro-architecture, we model each stage as a Directed Acyclic Graph (DAG), where each cell is a node of the DAG.

However, naively searching for the macro-architectures inevitably increases a large amount of additional parameters with corresponding computational overhead. More importantly, directly applying the method of cell search for searching the stages can lead to performance degradation, such as flattened stage structures. To address these issues, we carefully design three search rules and a depth loss, which allow us to systematically study the architectures at the stage level. First, we propose a novel yet simple method to search for the distribution of cells over different stages, under a constraint of computational complexity. This allows for a better optimization on the numbers of cells, which is never investigated in previous DARTS-series methods, where all stages are manually set to have the same number of cells. Second, with the optimized cell distribution computed in the previous step, we then focus on the search of the stage-level architecture. To the best of our knowledge, we are, for the first time, to explore the stage-level macro-architecture search, by relaxing the topological structures among different stages. We show that the proposed H-DAS can improve the performance for image classification prominently. The contributions of this work are summarized as follows:

- We propose a two-level Hierarchical Differentiable Architecture Search (H-DAS) that searches for structures at both the cell level and the stage level. H-DAS includes a cell-level search (H^c-DAS) and a stage-level search (H^s-DAS), for the micro- and macro-architectures, respectively.

- H^c-DAS is able to search for stage-specific cell structures within a greatly enlarged search space, comparing to that of DARTS. H^s-DAS includes a number of carefully-designed search rules and losses, which allows it to first search for an optimal distribution of cells over different stages, and then perform the search again to find the optimal structure for each stage.

- We conduct extensive experiments to demonstrate the effectiveness of the proposed H-DAS, which achieves a 2.41% test error on CIFAR10 and a 24.5% top-1 error on ImageNet. Moreover, the proposed stage-level search can be conducted based on the cell structures explored by other DAS methods, which further improves the performance.

Related Work

Neural architecture search (NAS) has recently been attracting more attentions, and it can be defined as searching for an optimal operation out of a defined operation set and the best connectivity between the operations, by using a Directed Acyclic Graph (DAG) (Zela, Siems, and Hutter 2020). The weight-sharing paradigm led to a significant improvement in search efficiency. Differentiable Architecture Search (DARTS) (Liu, Simonyan, and Yang 2018) relaxed the discrete search space to be continuous, making it possible to search architectures and learn network weights using gradient descent. P-DARTS (Chen et al. 2019) focused on bridging the depth gap between a search stage and an evaluation stage. PC-DARTS (Xu et al. 2019) performed a more efficient search without comprising the performance by sampling a small part of super-net to reduce the redundancy in a network space. In SmoothDARTS (Chen and Hsieh 2020), a perturbation-based regularization was proposed to smooth the loss landscape, and improve the generalizability. Fair-DARTS (Chu et al. 2020) solved the problem of aggregation of skip connections. These approaches follows the convention by stacking identical cells to form a chain-like structure.

Our stage-level search is related to that of (Liu et al. 2017), where low-level operations are assembled into a high-level motif, but the conceptions of states were not explored, and an evolutionary algorithm was applied for optimizing the search, making it much less efficient. In (Liang et al. 2019a), a computation reallocation (CR) was developed to search for the stage length for object detection, which inspired the current work, but our approach is more efficient by designing a hierarchical search that performs both cell-level and stage-level search jointly for a different target task.

Methodology

Preliminary. In this work, we follow the cell-level design of DARTS (Liu, Simonyan, and Yang 2018). The goal of DARTS is to search for a repeatable cell, which can be stacked to form a convolutional network. Each cell is a directed acyclic graph (DAG) of N nodes $\{x_i\}_{i=0}^{N-1}$, where each node can be represented as a network layer. Weighted by the architecture parameter $\alpha^{(i,j)}$, each edge (i,j) of DAG indicates an information flow from node i to node j, and is formulated as:

$$f_{i,j}(\mathbf{x}_i) = \sum_{o \in O_{i,j}} \frac{exp\{\alpha_{i,j}^o\}}{\sum_{o' \in O} exp\{\alpha_{i,j}^{o'}\}} o(\mathbf{x}_i). \qquad (1)$$

where $\mathbf{x}_i$ is the feature map at the i-th node, and o denotes candidate operations. More details, such as bi-level optimization, can be found in (Liu, Simonyan, and Yang 2018).

Hierarchical search space. In this work, we redesign the search space of DARTS-series methods, and propose a Hierarchical Differentiable Architecture Search (H-DAS) that enables the search both at the cell level (for micro-architecture) and at the stage level (for macro-architecture). As shown in figure 2, we relax the cell-level search space so that the network can learn the stage-specific cell structures
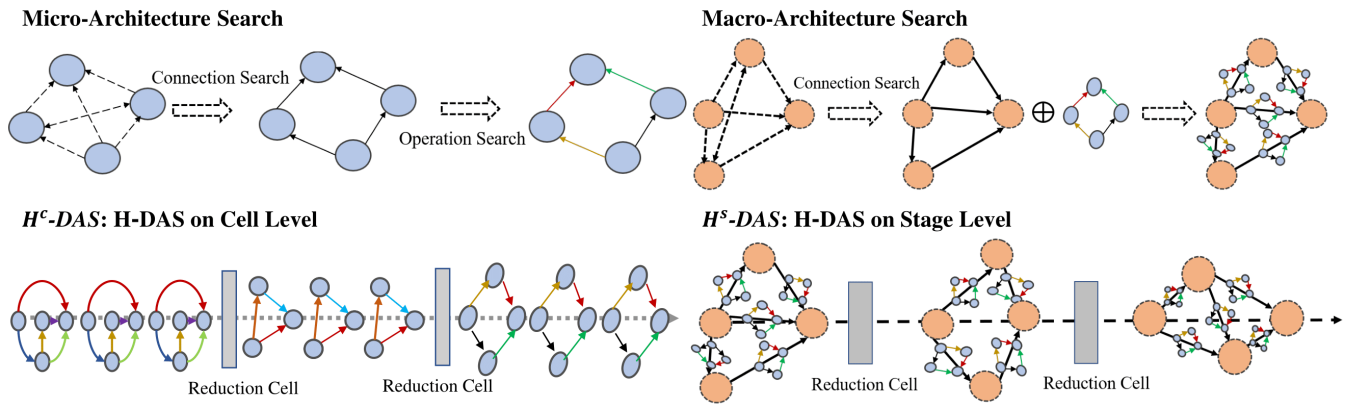
Figure 2: The overall pipeline of H-DAS, which includes a cell-level micro-architecture search (H^c-DAS) and a stage-level macro-architecture search (H^s-DAS) (best viewed in color). H^c-DAS searches for the structure of the cell, including operations (nodes) and connections between nodes. It relaxes the cell-level search space to learn *stage-specific* cell structures over different stages. H^s-DAS searches for the connections and operations between the cells in the stage-level search space.

for micro-architecture search. Then we model each stage as a DAG for macro-architecture search, which increases the variety of connections between cells. The two methods are named as H^c-DAS (for cell level) and H^s-DAS (for stage level), respectively.

Micro-Architecture

To enrich cell structures, we design H^c-DAS that relaxes the cell-level search space so that the networks can learn more meaningful stage-specific cell structures over different stages. In an extreme case, one can search for a specific structure for each cell, which results in a maximum of cell-level search space. However, it is not practical to performance NAS with such a large search space, which may make the NAS search process not stable, because the search can be influenced by many factors like hyper-parameters, the competition between model weights, and architecture parameters during bi-level optimization (Liang et al. 2019b). More importantly, it is difficult to set the depth of networks flexibly for different goals when the network has a unique structure per cell.

Search space relaxation. To relax the cell-level search space, and maintain the stability during the search period, our H^c-DAS aims to search for *stage-specific* cell-level structures, where the stages can have their own cell structures to capture different levels of semantics, while all cells in each stage has the same structure searched. Similar to DARTS-series methods, we use two reduction cells to divide the spatial resolution by 2 at 1/3 and 2/3 of the total depth of the networks, which separate the entire networks into three stages. The cell search involves both connections search and operation search. The goal of H^c-DAS is to find three structures of normal cells, each of which could be stacked repeatedly to form the optimal cell structure in the corresponding stage. The motivation of this design is that the shallow layers of CNNs often focus on learning low-level image information, like texture, while the deep layers of CNNs

pay more attention on high-level information, like semantic features. Therefore, the optimal cell structures at different stages should be diverse and play different functions, and our H^c-DAS naturally enriches the searched cell structures and increases the search space of the micro-architecture considerably.

Macro-Architecture

DARTS and its extensions mostly focus on searching for a repeatable cell structure, which is then stacked repeatedly and sequentially in a chain over multiple stages to form the networks. This setting assumes a simple chain-like structure for multiple stages, which significantly reduces the search space, and ignores the diversity in stage-level structures. Similarly, recent NAS methods based on mobile inverted bottleneck (Sandler et al. 2018), such as MnasNet (Tan et al. 2019), also stack the repeatable MBConvs sequentially to form the networks.

In addition to the search of cell structure, we found that it is important to build meaningful high-level macro-architecture by searching for the optimal connections between the cells over different stages. In this work, we introduce a Directed Acyclic Graph (DAG) structure to stage-level macro-architecture search. As shown in Figure 2, H^s-DAS searches for the connections between the searched (normal) cells in each stage to form a macro-architecture, which allows us to explore the power of stage-level structures. Notably, the searched macro-architectures can vary at different depths of networks, allowing the networks to learn meaningful cell connections at different stages.

However, searching for the macro-architecture is non-trivial, and would suffer from several problems. For example, many additional parameters and computational costs will be introduced by performing the search in the new stage-level search space. Moreover, the searched stage-level structures may become very shallow due to the ease on optimizing shallow networks, which will degrade the performance of networks. In this work, we carefully design
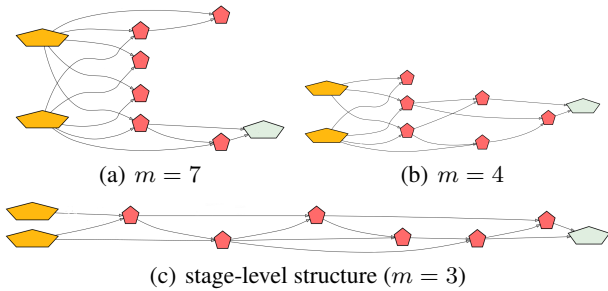
(a) $m = 7$　　　　(b) $m = 4$

(c) stage-level structure ($m = 3$)

Figure 3: Comparison on the number of candidate input cells. The stage level outside structure with N cells, and m is the number of previous candidate cells which the current cell could connect with, like a sliding window of m length, where $m \in [2, N + 1]$.

three important rules, with a novel depth loss, to ensure a robust and efficient search of the stage-level structure.

Rule 1: non-parametric connections. We design a set of candidate operations for the stage-level search space based on a key observation: the connections between cells may play a more important role than operation types. This observation is demonstrated by ablation experiments as shown in supplementary material (SM), and it was also discussed in (Xie et al. 2019). Hence, we define a small set of candidate operations, including *avg pooling*, *max pooling*, *skip-connect* and *None*, which do not have any learnable parameter, and therefore keep the network capacity similar to the conventional sequential structure. The search process can be formulated as Eq. (1), with a difference that $\mathbf{x}_i$ represents the output of the i-th cell. In this case, the conventional stage-level structure of DARTS can be considered as a special case of H^s-DAS, when only a single operation *skip-connect* is used between the cells.

Rule 2: stage output with selective cell aggregation. In the cell-level search, the final output of an entire cell is a depth-wise concatenation of the outputs of all nodes within the cell. In this setting, all nodes can contribute to the output in terms of computation. However, in the stage-level search by H^s-DAS, when the same concatenation strategy is adopted, the channel size of the output of each stage can be significantly increased, which in turn results in a great increase of the parameters. To minimize the additional parameters introduced by H^s-DAS, we compute the output of a stage by using the concatenated features of the last two cells in the stage. However, this rule introduces another issue that some intermediate cells are not directly connected to any subsequent cells, and hence are not included in the computational graph, e.g. Figure 3 (a). We alleviate this problem by introducing Rule 3 as follows.

Rule 3: constraint on preceding cells. A cell is defined as a dead cell when it is not connected to the stage output by any path in the graph. We empirically found that the

performance of the networks is negatively impacted by the number of dead cells, since the complexity and depth of the networks can be largely reduced when there are many dead cells, as shown in Figure 3. To alleviate this problem, we set a constraint on the number of preceding cells which each current cell can connect to. Namely, the cells as candidate predecessors can be considered as a sliding window for each cell in the stage. For example, a cell can only choose its preceding cells from three previous cells when we set $m = 3$ (Figure 3). This search rule can be considered as a trade-off between the stage-level search space and the number of active cells applied in the networks.

Depth loss. Without any restriction, the cells tend to directly connect to the input nodes, since shallower networks are generally easier to be optimized during the search (with respect to the network parameters), but often have lower performance. To alleviate this problem, we introduce a depth loss which takes into account the depth of the networks during the search. Each cell in the stage has a depth number, which indicates the number of intermediate cells between the current cell and input feature maps. We set the depth number of the input feature maps to be 0. In this case, the depth number of a cell is the weighted sum (i.e. α) of the depth numbers of its connected preceding cells. Therefore, we can calculate the depth number for each cell in a recursive manner:

$$L_{depth} = -\sum_{stage=1}^{3} \frac{1}{N} \sum_{i=0}^{N} \frac{1}{i} \sum_{j=0}^{i-1} \alpha_{i,j} \cdot (d_j + 1), \quad (2)$$

where $\alpha_{i,j}$ indicates the weight between node i and j, d_j is the depth of node j, and N is the number of cells in the stage. Minimizing the depth loss encourages the networks to go deep, and therefore improves the capability of networks.

Search for distribution of cells. With the three search rules and the new depth loss, we can now systematically study our stage-level architecture. First, we search for the number of cells in each stage. Previous DARTS-series methods mainly follow conventional configuration by manually setting the same number of cells for all stages, and it has not been verified whether such manual configuration is optimal. We therefore develop a simple method to explore the distribution of cells over different stages. The key idea of this search is to initialize an over-parameterized network (i.e. by containing more cells than necessary), and then remove the less impactful cells in each stage during the search, under a constraint of certain network capacity or FLOPs. Based on $Rule 2$, we introduce a parameter β to encode the importance of five pairs of adjacent cells that can connect to the output cell, among which only one pair will be selected at the end of the search. The output of a stage can be formulated as:

$$Output_{stage} = \sum softmax(\beta_{i,j}) \cdot concat(X_i, X_j) \quad (3)$$

where $\beta_{i,j}$ is the weight of output from cell i and cell j, and X_i are the output features of cell i. β is an architecture pa-

rameter that selects the one optimal pair of cells connecting to the final output of each stage, as illustrated in Figure 4. To incorporate β, E.q. 2 can be reformulated as:

$$L_{depth} = - \sum_{stage=1}^{3} \frac{1}{N} \sum_{i=0}^{N} \frac{1}{i} \sum_{j=0}^{i-1} \alpha_{i,j} \\ \times (d_j + 1) \times (1 + i \times softmax(\beta_{i-2})). \quad (4)$$

Crucially, a loss is adopted to constrain the computational complexity of the whole networks, and therefore the less important cells can be removed during the search. The cells in each stage share the same cell-level structure, and thus the computational complexity is constant for each stage. As a result, the loss with a constraint on the computational complexity can be simplified as follows:

$$L_{comp} = \sum_{i=0}^{S} \theta_i \sum_{j=0}^{N-N_{min}} softmax(\beta_{i,j}) \cdot (j + N_{min}), \quad (5)$$

where N is the number of cells in each stage, and N_{min} is the minimum number of cells in a stage. θ_i can be either multi-adds (FLOPs) or the number of parameters, depending on the desired constraint. We choose to have a constraint on the FLOPs. In this case, $\theta_i = 1$ since the stages share the same cell structure and thus the FLOPs of the cell are the same across three stages. The final loss function is:

$$Loss = L_{cls} + \delta L_{depth} + \gamma \times L_{comp}, \quad (6)$$

where δ and γ are weighting factors that balance the contributions of different losses. After training, we choose the pair of cells which has the largest β to connect to the output feature map. By setting the weights in E.q. 6, we can explore the number of cells in each stage, and obtain a network with a configurable total number of cells.

Notably, the weight of depth loss δ influences the final performance of the searched architecture. The optimal weight can be explored empirically. The corresponding experimental results regarding δ could be found in SM.

Search for stage-level structure. We fix the number of cells in each stage when searched, and then focus on searching for a stage-level architecture. The whole networks are constructed in the same manner as DARTS when the search is done. We can directly derive the network structure in the search of cells, but we empirically found that searching again with a fixed number of cells in each stage can improve the performance. This may be explained by a tighter search space which makes the search easier. With the relaxation of the topological structures among stages, we have a hierarchical structure that contains the search of nodes, cells and stages. The searched stage-level structure by H^s-DAS can be combined with other cell-level structures searched by existing DAS methods to further boost the performance.

Search space complexity. The proposed methods significantly increase the scale of search space. Specifically, H^c-DAS increases the search space of DARTS from 10^{18} to
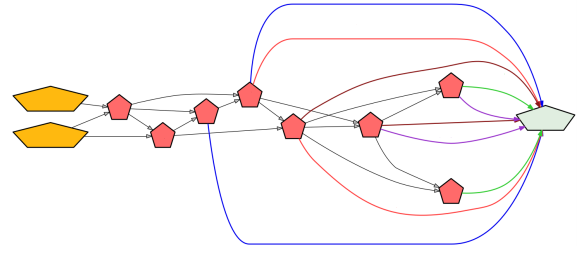
(a) Over-parameterized structure and candidate cells pairs
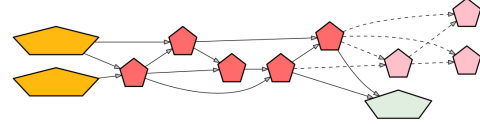
(b) Selection of an output-connected pair of cells

Figure 4: Search for distribution of cells in each stage by β.

10^{45} by allowing for stage-specific cells. Furthermore, by unchaining the conventional macro-architecture and searching for a specific macro-architecture for each stage, H^s-DAS has a search space of 10^{24} for a 20-cell network, without considering the cell-level search space. In the search of cell distribution by H^s-DAS, two additional cells are added in each stage, increasing the scale of the search space to 10^{33}, which is considerably larger than those in previous DAS methods. By including the cell-level search space, the full search space of a 20-cell H-DAS can reach to 10^{42}, which is significantly larger than previous methods, and naturally leads to a higher performance. The details of the complexity calculation are described in SM.

Experiments and Results

Implementation Details

We conduct experiments on CIFAR10 and ImageNet. In the search of cell-level structure, we follow DARTS (Liu, Simonyan, and Yang 2018) by using the same search space, hyperparameters and training scheme. We set $N_{min} = 4$ in Eq. (5). The stage-level search space contains 4 non-parametric operations which connect the cells, including: 3×3 *average pooling*, *skip connection*, 3×3 *max pooling*, *no connection (none)*. For training a single model, we use the same strategy and data processing methods as DARTS. More details can be found in SM.

Search for Cell Distribution

The search of cell distribution over three stages is performed under a constraint of certain computational complexity. It is interesting that different stages can have different numbers of cells at the beginning of the search or during the search, but the numbers of cells in all stages become the same at the end of the search. As discussed, adjusting the weighting factor γ in Eq. (6) leads to a different total number of cells in the networks. We therefore repeated the search several times with various values of γ, resulting in a different total number

Architecture	Test Err. (%)	Params (M)	Search Cost (GPU-days)	Search Method
ResNet (2016)	4.61	1.7	-	manual
DenseNet-BC (2017)	3.46	25.6	-	manual
NASNet-A (2018)	2.65	3.3	2000	RL
AmoebaNet-A (2019)	3.34	3.2	3150	evolution
Hierarchical evolution (2017)	3.75	15.7	300	evolution
ENAS (2018)	2.89	4.6	0.5	RL
Arch2Vec (2020)	2.56	3.6	100	BO
ProxylessNAS (2018)[†]	2.08	5.7	4	gradient
DARTS(2nd order) (2018)	2.76	3.3	1	gradient
SNAS (mild) (2018)	2.98	2.9	1.5	gradient
GDAS(FRC) (2019)	2.82	2.5	0.17	gradient
P-DARTS (2019)	2.62* / 2.50	3.4	0.3	gradient
PC-DARTS (2019)	2.57	3.6	0.1	gradient
NoisyDARTS (2020)	2.65* / 2.39	3.6	0.4	gradient
RDARTS (2020)	2.95	-	1.6	gradient
SDARTS-ADV (2020)	2.61	3.3	1.3	gradient
FairDARTS (2020)	2.54	3.3	0.41	gradient
ISTA-NAS (2020)	2.54	3.3	0.05	gradient
H^c-DAS	2.66	2.3	0.4	gradient
H^s-DAS	2.41	3.4	0.7$^\diamond$	gradient
H^s-DAS (with cell in P-DARTS)	2.30	3.5	0.3	gradient
H^s-DAS (with cell in NoisyDARTS)	2.34	3.6	0.3	gradient
H^s-DAS-autoAugment (2018)	1.99	3.6	0.7$^\diamond$	gradient

Table 1: Comparison with state-of-the-art architecture on CIFAR10. *: Our implementation by training the best cell architecture provided by the authors using the code of H-DAS. [†]: Obtained on a different search space with PyramidNet (Han, Kim, and Kim 2017) as the backbone. $^\diamond$: The search cost contains 0.4 GPU-day for cells and 0.3 GPU-day for stages.

of cells in the networks. But the cell distribution remains the same. Based on these observations, for a fair comparison, we set the number of cells in each stage in our H^s-DAS to be 6 for CIFAR10 and 4 for ImageNet, which are the same as other DARTS-series methods.

Results on CIFAR10

The results on CIFAR10 are compared in Table 1. H^c-DAS has many non-parametric connections in the cells of the last stage, and thus its parameter size becomes small, which is about 30% smaller than that of DARTS. However, it still has better performance than DARTS, suggesting that enlarging the cell-level search space and learning stage-specific cell-level structures could bring a great improvement. It also shows that searching for a single repeatable cell structure to form a chain-like network is not fully optimized.

Furthermore, by combining with the cell structures searched by recent excellent DAS methods (Chen et al. 2019; Chu, Zhang, and Li 2020), our H^s-DAS, which searches for more meaningful cell connections over different stages, can improve the stage-of-the-art methods by achieving an error of 2.30% on CIFAR10, which is the best among DAS methods. Notice that ProxylessNAS has a better result, but with 60-100% more parameters and ×10 search time comparing to other methods, including our H^s-DAS.

We also found that the performance of H^s-DAS is better than H^c-DAS, suggesting that the macro-architecture plays an important role in neural architecture. A simple chain-like sequential structure would significantly limit the search

space, and discard the importance of stage-level structures. It is beneficial to unchain the search space and pay more attention on developing more meaningful macro-architectures in the future research.

Results on ImageNet

To verify the generalization of our searched structures, we train our networks on ImageNet using the architecture searched from CIFAR10. Results of our H^c-DAS and H^s-DAS on ImageNet are compared in Table 2. We transfer the cell-level and stage-level structures with four normal cells searched from CIFAR10 to ImageNet. Again, H^s-DAS outperforms H^c-DAS with a large margin, demonstrating the superiority of meaningful cell connections in stage-level structures. Additionally, our H^s-DAS is comparable against other state-of-the-art DAS methods, without using the advanced network block design like MBConvs.

Ablation Study

To demonstrate the effectiveness of cell-level search and stage-level search, we do ablation experiments on CIFAR10 and ImageNet with various configurations. The details are reported in Table 3. We can summary that, (1) our H^c-DAS outperforms DARTS by allowing for searching for different cell structures over different stages; (2) comparing to DARTS, our H^s-DAS* which performs the stage-level search, can improve the performance of the unsearched chain-like structure; (3) H^s-DAS can further improve H^s-DAS* by implementing the *stage-specific* macro-

Architecture	Test Err.(%)		Params (M)	×+ (M)	Search Cost (GPU-days)	Search Method
	top-1	top-5				
Inception-v1 (2015)	30.2	10.1	6.6	1448	-	manual
MobileNet-v2 (2018)	25.3	-	6.9	585	-	manual
ShuffleNet 2x (v2) (2018)	25.1	-	7.4	591	-	manual
NASNet-A (2018) [††]	26.0	8.4	5.3	564	1800	RL
AmoebaNet-C (2019) [††]	24.3	7.6	6.4	570	3150	evolution
PNAS (2018) [††]	25.8	8.1	5.1	588	225	SMBO
MnasNet-92 (2019) [††]	25.2	8.0	4.4	388	-	RL
MobileNet-v3-large (2019)[††]	24.8	-	5.4	219	-	RL
DARTS (2nd order) (2018) [†]	26.7	8.7	4.7	574	1	gradient
GDAS (2019) [†]	26.0	8.5	5.3	581	0.21	gradient
SNAS (mild) (2018) [†]	27.3	9.2	4.3	522	1.5	gradient
P-DARTS (2019)[†]	25.1* / 24.4	7.7* / 7.4	4.9	557	0.3	gradient
SinglePath-NAS (2019)[††]	25.0	7.8	-	-	0.15	gradient
ProxylessNAS (GPU) (2018)[††]	24.9	7.5	7.1	465	8.3	gradient
PC-DARTS (2019) [†]	25.1	7.8	5.3	586	0.1	gradient
RandWire-WS (2019) [††]	25.3	7.8	5.6	583	-	random
SDARTS-ADV (2020) [†]	25.2	7.8	-	-	1.3	gradient
FairDARTS (2020) [†]	24.9	7.5	4.8	541	0.4	gradient
ISTA-NAS (2020) [†]	25.1	7.7	4.78	550	2.3	gradient
H^c-DAS [†]	25.9	8.4	5.0	578	0.5	gradient
H^s-DAS [†]	24.5	7.7	5.1	572	0.3	gradient

Table 2: Comparison with state-of-the-art architectures on ImageNet (mobile setting). *: Our implementation by training the best architecture provided by the authors using the code of H-DAS. [†]: Searched on CIFAR10. [††]: Searched on ImageNet.

Search Space	Cell		Stage		Test Err.(%)	
	Sh.	Spec.	Sh.	Spec.	C10	ImgNet
DARTS	✓	✗	-	-	2.76	26.70
H^c-DAS	✗	✓	-	-	2.66	25.91
H^s-DAS*	✓	✗	✓	✗	2.58	25.45
H^s-DAS[†]	✗	✓	✗	✓	2.51	25.80
H^s-DAS	✓	✗	✗	✓	2.30	24.50

Table 3: Performance of the cell-level search and stage-level search, by using the *shared* structures over all stages or the *stage-specific* architectures for different stages. DARTS and H^c-DAS do not perform stage-level search.

Structure	m	# Dead cells	# Depth	Test Err.(%)
Fig.3(a)	7	4	2	3.08
Fig.3(b)	4	1	3	2.91
Fig.3(c)	3	0	5	2.58

Table 4: Constraint of input cells. The three macro-architectures are retrained on CIFAR10 with the same configuration. 'm': the length of sliding window. '# Dead cells': the number of cells which are not connected to a subsequent cell. '# Depth': the number of cells in the longest path from input to output.

architecture search; (4) it is interesting that H^s-DAS† has a lower performance than H^s-DAS by performing the *stage-specific* cell search. This would result in a larger search space which might be over-complex for a search algorithm to learn an optimal result.

Additionally, as we found empirically in Section 3.2 that the number of dead cells in the searched architecture would impact the performance of the networks. We further verify it by training the rest two structures in Figure 3 on CIFAR10. As presented in Table 4, a network with less dead cells and a deeper macro-architecture obtains a higher performance.

Furthermore, to show the effectiveness of our searching strategy on macro-architecture, we keep the same cell-level structures and compare the macro-architectures searched with a random baseline. By following (Yang, Esperança, and Carlucci 2019), we calculate a relative improvement over this random baseline as $RI = 100 \times (Acc_m - Acc_r)/Acc_r$,

which provides a quality measurement of the search strategy. Acc_m and Acc_r indicate the top-1 accuracy of a search method and the random sampling strategy, respectively. Our H^s-DAS achieves a RI of 0.44, with a significant improvement over 0.32 of DARTS.

Conclusions

We have presented our Hierarchical Differentiable Architecture Search (H-DAS) which preforms both cell-level and stage-level architecture search. In the cell-level search, H-DAS improves DARTS baseline by exploring stage-specific cell structures. Importantly, we formulate the stage-level structure as a directed acyclic graph, which allows us to search for a more advanced architecture than the conventional chain-like configuration. Our two-level search enlarges the search space considerably, and improves the performance of previous DARTS-series methods significantly.

References

Baker, B.; Gupta, O.; Naik, N.; and Raskar, R. 2016. Designing neural network architectures using reinforcement learning. *arXiv preprint arXiv:1611.02167* .

Cai, H.; Zhu, L.; and Han, S. 2018. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332* .

Chen, X.; and Hsieh, C.-J. 2020. Stabilizing Differentiable Architecture Search via Perturbation-based Regularization. *ICML* .

Chen, X.; Xie, L.; Wu, J.; and Tian, Q. 2019. Progressive Differentiable Architecture Search: Bridging the Depth Gap between Search and Evaluation. *arXiv preprint arXiv:1904.12760* .

Chu, X.; Zhang, B.; and Li, X. 2020. Noisy Differentiable Architecture Search. *arXiv preprint arXiv:2005.03566* .

Chu, X.; Zhou, T.; Zhang, B.; and Li, J. 2020. Fair darts: Eliminating unfair advantages in differentiable architecture search. *ECCV* .

Cubuk, E. D.; Zoph, B.; Mane, D.; Vasudevan, V.; and Le, Q. V. 2018. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501* .

Dong, X.; and Yang, Y. 2019. Searching for a robust neural architecture in four gpu hours. In *CVPR*.

Han, D.; Kim, J.; and Kim, J. 2017. Deep pyramidal residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5927–5935.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*.

Howard, A.; Sandler, M.; Chu, G.; Chen, L.-C.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; et al. 2019. Searching for mobilenetv3. In *ICCV*.

Huang, G.; Liu, Z.; Van Der Maaten, L.; and Weinberger, K. Q. 2017. Densely connected convolutional networks. In *CVPR*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*.

Liang, F.; Lin, C.; Guo, R.; Sun, M.; Wu, W.; Yan, J.; and Ouyang, W. 2019a. Computation reallocation for object detection. *arXiv preprint arXiv:1912.11234* .

Liang, H.; Zhang, S.; Sun, J.; He, X.; Huang, W.; Zhuang, K.; and Li, Z. 2019b. Darts+: Improved differentiable architecture search with early stopping. *arXiv preprint arXiv:1909.06035* .

Liu, C.; Zoph, B.; Neumann, M.; Shlens, J.; Hua, W.; Li, L.-J.; Fei-Fei, L.; Yuille, A.; Huang, J.; and Murphy, K. 2018. Progressive neural architecture search. In *ECCV*.

Liu, H.; Simonyan, K.; Vinyals, O.; Fernando, C.; and Kavukcuoglu, K. 2017. Hierarchical representations for efficient architecture search. *arXiv:1711.00436* .

Liu, H.; Simonyan, K.; and Yang, Y. 2018. Darts: Differentiable architecture search. *arXiv:1806.09055* .

Ma, N.; Zhang, X.; Zheng, H.-T.; and Sun, J. 2018. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *ECCV*.

Pham, H.; Guan, M. Y.; Zoph, B.; Le, Q. V.; and Dean, J. 2018. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268* .

Real, E.; Aggarwal, A.; Huang, Y.; and Le, Q. V. 2019. Regularized evolution for image classifier architecture search. In *AAAI*.

Real, E.; Moore, S.; Selle, A.; Saxena, S.; Suematsu, Y. L.; Tan, J.; Le, Q. V.; and Kurakin, A. 2017. Large-scale evolution of image classifiers. In *ICML*.

Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; and Chen, L.-C. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*.

Simonyan, K.; and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* .

Stamoulis, D.; Ding, R.; Wang, D.; Lymberopoulos, D.; Priyantha, B.; Liu, J.; and Marculescu, D. 2019. Single-path nas: Designing hardware-efficient convnets in less than 4 hours. *arXiv preprint arXiv:1904.02877* .

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *CVPR*.

Tan, M.; Chen, B.; Pang, R.; Vasudevan, V.; Sandler, M.; Howard, A.; and Le, Q. V. 2019. Mnasnet: Platform-aware neural architecture search for mobile. In *CVPR*.

Xie, S.; Kirillov, A.; Girshick, R.; and He, K. 2019. Exploring randomly wired neural networks for image recognition. *arXiv preprint arXiv:1904.01569* .

Xie, S.; Zheng, H.; Liu, C.; and Lin, L. 2018. SNAS: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926* .

Xu, Y.; Xie, L.; Zhang, X.; Chen, X.; Qi, G.-J.; Tian, Q.; and Xiong, H. 2019. Pc-darts: Partial channel connections for memory-efficient differentiable architecture search. *arXiv preprint arXiv:1907.05737* .

Yan, S.; Zheng, Y.; Ao, W.; Zeng, X.; and Zhang, M. 2020. Does unsupervised architecture representation learning help neural architecture search? *Advances in Neural Information Processing Systems* 33.

Yang, A.; Esperança, P. M.; and Carlucci, F. M. 2019. NAS evaluation is frustratingly hard. *arXiv preprint arXiv:1912.12522* .

Yang, Y.; Li, H.; You, S.; Wang, F.; Qian, C.; and Lin, Z. 2020. Ista-nas: Efficient and consistent neural architecture search by sparse coding. *Advances in Neural Information Processing Systems* 33.

You, S.; Huang, T.; Yang, M.; Wang, F.; Qian, C.; and Zhang, C. 2020. GreedyNAS: Towards Fast One-Shot NAS with Greedy Supernet. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1999–2008.

Zela, A.; Elsken, T.; Saikia, T.; Marrakchi, Y.; Brox, T.; and Hutter, F. 2020. Understanding and Robustifying Differentiable Architecture Search. *ICML* .

Zela, A.; Siems, J.; and Hutter, F. 2020. NAS-Bench-1Shot1: Benchmarking and Dissecting One-shot Neural Architecture Search. *arXiv preprint arXiv:2001.10422* .

Zhong, Y.; Deng, Z.; Guo, S.; Scott, M. R.; and Huang, W. 2020. Representation Sharing for Fast Object Detector Search and Beyond. *arXiv preprint arXiv:2007.12075* .

Zoph, B.; and Le, Q. V. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578* .

Zoph, B.; Vasudevan, V.; Shlens, J.; and Le, Q. V. 2018. Learning transferable architectures for scalable image recognition. In *CVPR*.