ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CVL Computer Vision Lab

Efficient ConvNet Architectures
Differentiable Neural Architecture Search

Semester Thesis

Silvio Paganucci
Department of Information Technology and Electrical Engineering

Advisor: Dr. Radu Timofte & Yawei Li
Supervisor: Prof. Dr. Luc Van Gool

February 25, 2020

Abstract

Despite all benefits that come with neural networks, designing a suitable architecture for a given problem is crucial for its final performance. A few years ago, there was no workaround for creating neural architectures than using expertise to manually design suitable networks, which is known to be a time-consuming and error-prone process. Recent works aimed to automatize this process by introducing the area of neural architecture search (NAS). In this project, we lay focus on the method of differentiable architecture search (DARTS). It has proven to produce well performing architectures in comparatively small search-time. We analyze existent DARTS methods to get a better understanding of the significance of used settings, as well as examine different approaches to improve the results in at least one of the following metrics: Search-time, validation-accuracy or number of parameters.

Acknowledgements

I want to thank Prof. Dr. Luc Van Gool and Dr. Radu Timofte for letting me have the opportunity to do this project under their supervision. Also a special thank to Dr. Radu Timofte and Yawei Li for supporting me during my project and for giving me constructive inputs when necessary.

Lastly, my thanks go to the Computer Vision Lab as well, for providing me access to their GPU cluster. Without, conducting my experiments would have become impossible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Three main groups of search methods emerged in the area of neural architecture search (NAS). For one there are evolutionary and reinforcement learning based methods. They're both capable of producing well performing architectures, but have a huge drawback when it comes to search-time, since consecutive training and evaluation of different architectures may take more than 1000 GPU days. The third and newest method, called DARTS (Differentiable ARchiTecture Search) [6], uses a different approach to reduce search-time to just a single GPU day. While in earlier methods, architecture search has been treated as a black-box optimization problem over a discrete domain, DARTS relaxes the search space to be continuous, thus making it possible to use gradient-based optimization with respect to the architecture's validation set performance. It does not involve any controllers or hyper-networks for performance estimation, which makes it much simpler than many existing approaches, and yet produces competitive performance with the state of the art on classification tasks.

DARTS aims to learn high-performance building blocks within a defined search space, as suggested in [8]. Instead of searching for the whole neural architecture at once, the problem scales down to a cell search problem where two different cell types, the normal- and reduction cell, form the fundamental elements. To build the final architecture, we stack the cells to blocks as visualized in figure 1.1. The cells themselves consist of input, intermediate and output nodes. The output of each cell is forwarded unchanged to its two successor cells, where they form the first or the second input respectively. Intermediate nodes have two input

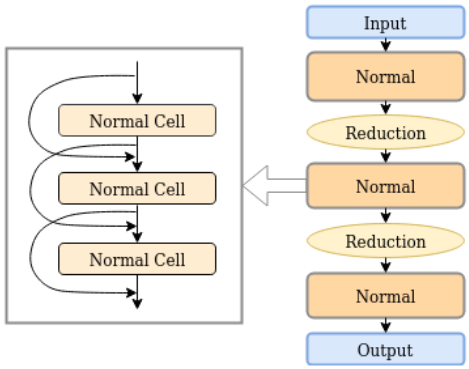


Figure 1.1: Block structure of network architecture

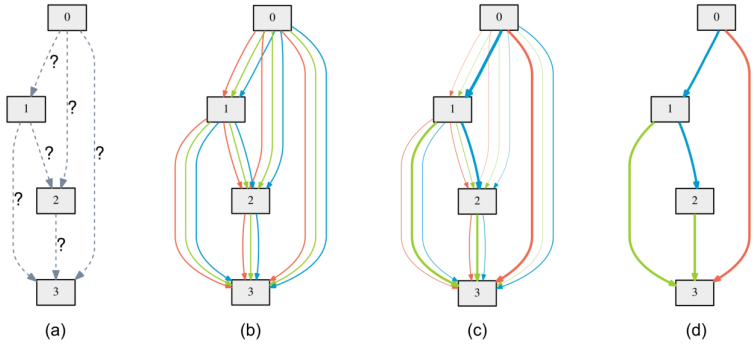


Figure 1.2: Overview of architecture search with DARTS [6]

connections coming from either the inputs or any predecessor node within the cell. These connections perform a specific operation on the inputs (data at origin of connections), whereby the results get concatenated together to form the 'state' of the node. Finally, the states of all intermediate nodes get concatenated to get the only output of the cell. The structure results in a directed acyclic graph (DAG), an example cell can be seen in Figure A.4.

To search for cell structures we need a predefined search space containing different operations such as convolution, pooling, etc. Since the operations on the edges are initially unknown, we put all the candidate operations within the search space on all possible/valid edges of the cells. By optimizing a bi-level optimization problem, training of network weights as well as tightening the relaxed mixture probabilities of the candidate operations, we get towards an optimal operation choice for each edge. At the end of the search process, the operation with the highest probability on each edge wins and is picked as the final choice. An overview of the relevant steps is contained in 1.2.

1.1 Focus of this Work

DARTS is able to handle convolutional and recurrent architectures. Its performance is evaluated on the data sets CIFAR-10, CIFAR-100 and ImageNet for convolutional, and on Penn Treebank for recurrent neural networks. For simplicity reasons, we restrict ourselves to evaluate our experiments on CIFAR-10 only. Further is not only the original implementation of DARTS worked on, but also a derived version of the method called progressive differentiable architecture search (PDARTS) [1]. Its benefits lay in cutting down the search-time from 1 to 0.3 GPU-days, making it possible to run more experiments within shorter time, while also achieving a better test-error of 2.50% on CIFAR-10 compared to 2.83% with original DARTS (numbers according to [1]).

The focus of this project lays on having a closer look at the settings used in the search algorithm. For one, there's a fixed search space, restricting the search to only find optimal solutions within this space. In this work, we analyze the importance of currently present operations and propose an additional operation to amend the search space. Further, there's a big loss in terms of freedom when building the network using cells and blocks. We study the relevance of restricting the network to exactly one normal and one reduction cell, both having 4 intermediate nodes. For that purpose we try on one hand to restrict the degree of freedom even

further by reducing the number of intermediate nodes to 3, and on the other hand giving it some freedom back by introducing multiple normal cells, a distinct one for each of the 3 blocks. And last but not least, we have a look at the discretization step at the end of the search process and propose and alternative strategy to select the best operation for the final network. More details on that in chapter 3.

1.2 Thesis Organization

A short overview of all aspects that have been pursued in this project:

- Reducing the degree of freedom by restricting the number of intermediate nodes from 4 to 3. (Evaluation on accordingly modified PDARTS)
- Increasing the degree of freedom by introducing 2 more normal cells, leading to a total of 3 distinct normal and 1 reduction cell. (Evaluation on accordingly modified PDARTS)
- Analyzing the importance of each operation within the used search space. (Evaluation on both, DARTS and PDARTS)
- Expanding the search space with an additional operation. (Evaluation on PDARTS)
- Introducing a new discretization method for selecting the best operations for each edge at the end of the architecture search. (Evaluation on accordingly modified PDARTS)

Chapter 2

Related Work

Since the publication of the DARTS paper [6] in 2018, many papers followed to addressed specific problems/limitations in the setting of DARTS and proposed solutions or improvements to further cut down search-time or increase validation accuracy. A paper published in September of 2019 [5] addressed a problem they call the 'collapse' of performance when the number of search epoch becomes too large. Reasoning with the insertion of too many skip-connections (parameter free forwarding/identity operation) with progressing epochs. They propose an algorithm DARTS+ with a simple solution, namely introducing an early-stopping criterion which stops optimizing when the criterion is met, producing architectures with an upper bounded number of skip-connections. Others suggest to improve the search procedure by sampling only a small part of the super-net in search phase to reduce the redundancy in the network space [3], [7]. Facing the problem of discretization of the searched architecture, FairDARTS [2] proposes the addition of a 0/1-loss to push architecture weights towards 0 or 1 and thus lower the gap between continuous search space and discrete architecture. And there is sharpDARTS [4], which improved the learning rate scheduler for final training using SGD and also proposed some modifications in the search space.

We can see that there are yet many things to improve. Many approaches tried to make the original DARTS method more efficient when it comes to search-time, others aimed to lower the variance and produce more reliable solutions while producing well performing architectures. But what they have all in common, they mostly stick to the presented search space, as well as the presented final architecture setup from the original DARTS paper. In this project, we thus analyze the importance of operations within the presented search space and propose the integration of a new operation. In addition, we have a look at the cell-block-nature of the final architecture and experiment with its restrictions on variance. In our last matter, we come closer to a problem also addressed by other papers (as [2]), which tackles the discretization gap and its consequences in terms of reliable solutions.

Chapter 3

Materials and Methods

3.1 Preliminary: DARTS

The DARTS method, as proposed in [6], is used as baseline for this project. It aims to search for robust cells of type normal and reduction. A cell represents a directed acyclic graph (DAG) of N nodes, $\{x_0, x_1, ..., x_{N-1}\}$, where each node can be seen as a network layer. The search space (or operation space) is denoted as Ω, wherein each candidate operation is represented by o($\cdot$). An edge $E_{(i,j)}$, connecting node x_i to node x_j, consists of a set of operations, each weighted by the architecture parameters $\alpha^{(i,j)}$. The state (data content) of a node is calculated over equation 3.1,

$$f_{i,j}(x_i) = \sum_{o \in \Omega_{i,j}} \frac{exp(\alpha_o^{(i,j)})}{\sum_{o' \in \Omega} exp(\alpha_{o'}^{(i,j)})} o(x_i) \tag{3.1}$$

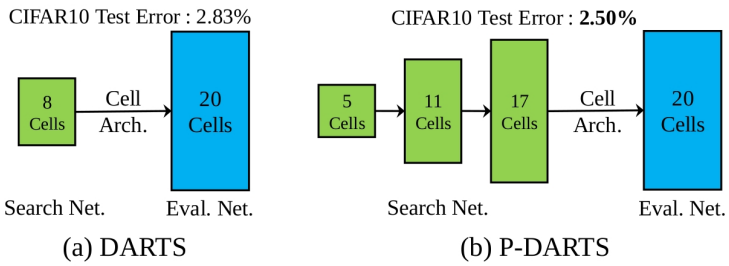
where $i < j$, thus only allowing connections from lower to higher indexed nodes. x_0 and x_1 form the inputs of the cell, intermediate nodes can be represented as $x_j = \sum_{i<j} f_{i,j}(x_i)$ and the cell output results from $x_{N-1} = concat(x_2, x_3, ..., x_{N-2})$, where the states of all intermediate nodes get concatenated in the channel dimension. Refer to the original DARTS paper [6] for more details.

3.2 Preliminary: PDARTS

3.2.1 Search Architecture

The original DARTS implementation performs its architecture search on a fixed number of 8 cells (6 normal- and 2 reduction cells). Search may thus result in optimal solutions for the shallow network, but does not necessarily be optimal for the deeper network of final evaluation, consisting of 20 cells for CIFAR-10 (18 normal- and 2 reduction cells). In [1], they address this problem under the name depth gap between search and evaluation.

They propose an alternative search algorithm PDARTS to lower the depth gap by cutting the search phase down into 3 stages. Starting with a shallow network of just 5 cells (3 normal- and 2 reduction cells), it already excludes part of the operation-options of the search space for each edge by kicking out the ones with the lowest probabilities. Having a smaller search space for each edge, we can insert more normal cells into the network for the next stages without increasing the time necessary for search. This allows to get closer to the final architecture of 20 stacked cells, allowing a better estimate of optimal solution during search. The



(a) DARTS (b) P-DARTS

Figure 3.1: Comparison between search structure of DARTS and PDARTS [1]

principle is visualized in Figure 3.1. For a more detailed overview, have a look at A.1 or refer to the original PDARTS paper [1].

3.2.2 Search Space Regularization

Critical for the performance of searched architectures is the number of included skip-connection operations. Similar to [5], they restrict the final network to have a fixed number of skip-connections within normal cells. After completed search, the algorithm replaces present skip-connections one by one, starting with the lowest weighted and substituting them with the next best weighted, non-skip operation. The user can now select a network with his preferred number of skip-connections. They recommended to use 2 skip-connections, since this yielded the best results during final evaluation.

3.3 Modified Architecture Structure

DARTS or derived methods all share similar architectural structure. Their fundamental elements include 2 distinct cell types, one normal- and one reduction cell. Within each of these cells, there are 4 intermediate nodes representing the layers of the network. The repeating nature of this structure restricts the architecture from developing depth-specific optimal sub-structures. We call this the limitation of the degree of freedom, since it bounds the possible variance within the network.

3.3.1 Intermediate Nodes

For exploring the structure space of network architectures we first tighten this restriction by lowering the number of intermediate nodes from 4 to 3. We denote the cell structure as a DAG $\{x_0, x_1, ..., x_{N-1}\}$, where x_0 and x_1 represent the inputs and x_{N-1} the output. While originally we optimized over $\#weights_{arch} = count(\alpha) * size(\Omega)$ architecture weights, whereby $count(\alpha) = \sum_{i=2}^{N-2} i$, we reduce the number of optimization parameters by $(N-2) * size(\Omega)$ per cell with removing one intermediate node. In our case, we end up with $\#weights_{arch} = 9 * size(\Omega)$ for 3, while originally we had $\#weights_{arch} = 14 * size(\Omega)$ for 4 intermediate nodes. This corresponds to a relative decrease of approximately $\frac{1}{3}$ in number of weights, thus resulting in a simplified search setting, which should demand less time for architecture search.

Since we reduce the number of per-cell operations by 2 and also get smaller dimensions in the output nodes of each cell (concatenation of all intermediate nodes), we end up with less trainable parameters in the final network. For fair comparison of the results, we need to compensate for this loss. For that reason, we

try both, increasing the number of initial channels, as well as increasing the number of layers (number of normal cells per block) to match the desired parameter number.

3.3.2 Normal Blocks

On the other hand, we try to increase the variance within the network by introducing 2 new cell types which we call normal2- and normal3 cell. These types form the basis of the normal blocks as visible in Figure 3.2. The purpose behind this is to increase the degree of freedom for final architectures and get a depth specific information extraction.

One might think this increases the computational effort for architecture search by much, but when having a closer look we see that this isn't the case. As stated in 3.3.1, the number of architecture weights calculates by $\#weights_{arch} = count(\alpha) * size(\Omega)$ with $count(\alpha) = \sum_{i=2}^{N-2} i$. With regards to this number, we increase the number of weights to optimize over by a factor of 2. But since we perform a bi-level optimization, this forms only one side of the computational effort. The main part, the optimization of operation specific parameters for all operations in Ω, remains the same. And since $\#weights_{arch} \ll \#weights_{ops}$, the additional cost for search time is negligible.

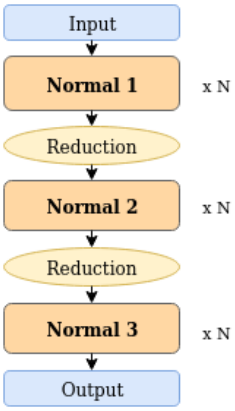


Figure 3.2: Network structure with 3 distinct normal blocks

3.4 Search Space Analysis

The search space Ω, as used in DARTS [6], consist of the following 8 candidate operations:

- none/zero
- 3x3 max pooling
- 3x3 average pooling
- skip-connection
- 3x3 separable convolution
- 5x5 separable convolution
- 3x3 dilated convolution
- 5x5 dilated convolution

We analyze the importance of the operations by running different architecture searches while excluding one operation in each run. Our findings farther led to the conduction of a pursuing analysis by excluding multiple operations at a time.

3.4.1 Imbalance of Convolutions

While examining the implementations behind the operations, we noticed something worth to have a look at. For dilated convolutions, we perform the following series of logical operations:

$$ReLU \rightarrow Conv_{dil} \rightarrow Conv_{1x1} \rightarrow BatchNorm$$

whereas for separable convolutions, the series looks as follows:

$$ReLU \rightarrow Conv \rightarrow Conv_{1x1} \rightarrow BatchNorm \rightarrow ReLU \rightarrow Conv \rightarrow Conv_{1x1} \rightarrow BatchNorm$$

Having different depths for separable and dilated convolutions may lead to an advantage or disadvantage for one of them during architecture search. We try to eliminate this imbalance by increasing the depth of dilated convolution operations. For one, by performing the same block of logical operations of dilated convolution two times, and second, by performing a block of regular, separable convolution after the dilated convolution block. Latter one being in principle the same approach as proposed in sharpDARTS [4].

3.4.2 New Operation

Reasoned with the findings described in 3.4.1, we expand Ω by a new operation type. Since the depth of operations might have an effect on getting selected for the final architecture, we introduce a new operation named 'Deep Separable Convolution'. Following the structure described in 3.4.1, deep separable convolution performs 2 separable convolutions in a row. This results in a much higher number of parameters, we thus halve the channel-sizes of intermediate operations (Halve channels at first 1x1 convolution, double at last 1x1 convolution for final output).

Appending a 3x3 and 5x5 version to Ω also increases the cost for architecture search, as we now have more candidate operations for each edge. In first experiments, we neglect this issue to analyze the effect of the newly introduced operation. In latter ones, we also examine the replacement of the 5x5 separable convolution with the new 3x3 deep separable convolution, reasoning with well performing architectures from 3.3.1, where no 5x5 separable convolutions could be found in the final architectures.

3.5 Discretization by Grouping

A known bottleneck of differentiable architecture search is the discretization from the relaxed search space to the final architecture. After completed search, we have a weighted distribution of how much each operation accounts to the final performance. Picking the operations with the biggest weights seems reasonable, but may be sub-optimal since we have a mixed influence of a whole set of operations. EDAS [3] tackled this problem by randomly sampling edges of the network to optimize, while keeping only the top-ranked operation active on all the other edges. This way, the architecture is optimized with regards to the final operation selection.

We present another approach, only affecting the operation selection at the end of architecture search. By having a look at the search space Ω (as in 3.4), we notice that there are operations that are related to one another and may thus have a collective influence on $f_{i,j}(x_i)$, even if they're not individually weighted the highest (see equation 3.1). Using this aspect, we assign each operation within Ω to a specific group. For the scope of this work, we propose a very simple grouping strategy that divides operations into the following groups: $none$, $skip$, $pool$ or $conv$. We now add two steps in front of the final architecture selection. In the first step, we pick the group with the highest score for each edge $argmax(\sum_{o \in group(\Omega_{i,j})} \alpha_o^{(i,j)})$. In the second step, we assign the group-score to the highest weighted operation within the group. The final operation selection for each node is now performed as before, but using the updated weights on each edge.

Chapter 4

Experiments and Results

We conducted our experiments on the classification data-set CIFAR-10. It consists of 60k RGB images with a resolution of 32x32 and is split into a training and testing set by a ratio of 5:1. A different split-ratio is used for architecture search, where half of the data is used for tuning the operation parameters, while the other half is used for tuning the architecture (operation weights).

All experiments were conducted with the original settings of the according implementation, whereat changes are stated where made. So, unless otherwise indicated, refer to the DARTS [6] or PDARTS [1] paper, respectively, for more technical details on the experiment setup. We used the code available at [1] for DARTS, and [2] for PDARTS. (Note that we didn't use the original code of the DARTS paper, but a version implemented with pytorch 1.0)

Results have to be interpreted with caution, since limited computational resources didn't allow extensive evaluation and mostly restrict to one single final training.

4.1 Modified Architecture Structure

For the first experiments of this section, we made according changes in the PDARTS implementation to support user definable numbers of intermediate nodes. For the second part, we expanded the network structure of PDARTS to support up to 4 different cell types, 3 normal- and 1 reduction cell.

4.1.1 Intermediate Nodes

Using the modified PDARTS algorithm, we search for architectures containing 3 intermediate nodes within each cell, all other settings remain unchanged. We perform searches with batch sizes of 32 and 64, where we achieve with the latter setting a higher validation accuracy at the end of search with 3 compared to the original 4 nodes. The results are visible in Table A.1. For each of the listed settings, 3 searches were performed. We observe a relative decrease in search cost of 32.5% in terms of time.

For obvious reasons, we get sparser genotypes for architectures with just 3 intermediate nodes, which would result in networks with a smaller number of trainable parameters. To make a fair comparison in terms of accuracy, we try two approaches. First, we increase the number of normal cells within each normal block, denoted as layers, resulting in a deeper network. Since the network contains a total number of 3 normal blocks, we insert additional cells only by a number divisible by 3 to maintain overall structure. In

[1] https://github.com/khanrc/pt.darts

[2] https://github.com/chenxin061/pdarts

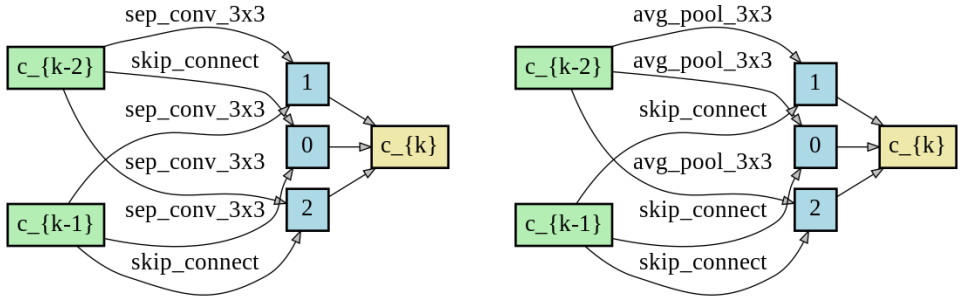


Figure 4.1: Genotype of PDARTS-3b

the second approach, we increase the number of initial channels of the network to match the desired number of trainable parameters. We run final training with 600 epochs of SGD with momentum and a learning rate decay following cosine annealing (unchanged settings). An overview of conducted evaluations is contained in Table 4.1.

genotype	parameter size [M]	layers	init channels	batch size	lr	val accuracy [%]
PDARTS[3]	3.43	20	36	96	0.025	97.54
				64	0.015	97.45
PDARTS	3.65 ± 0.05	20	36	96	0.025	97.43 ± 0.04
PDARTS-3a	3.33	29	36	96	0.025	97.46
	3.37	20	44			97.29
PDARTS-3b	3.36	26	36	64	0.015	97.43
	3.44	20	42			97.39

Table 4.1: Evaluation of architectures containing sparser cells

First we notice that, compared to best genotype from the paper [1], consecutive search and evaluation with standard settings of the algorithm results in larger final networks while having a best validation accuracy bellow 97.50% (averaged over 2 runs).

We can further see in Table 4.1, that a deep network with additional layers does perform better than the same one with increased initial channel sizes (see Figure A.2 for evaluation comparison). Whereby the networks PDARTS-3a&b with increased number of layers come close to the performance of the best found genotype of PDARTS. In Figure A.2, we notice a slower convergence rate for the deeper network compared to the one with increased initial channels. We thus perform a pursuing evaluation of PDARTS-3b where we used a batch size of 96 and increase the number of epochs from 600 to 1000 to ensure convergence (see Figure 4.2), resulting in a validation accuracy of 97.56% that even outperforms the published genotype of [1]. The genotype of PDARTS-3b can be seen in Figure 4.1. Also to mention that it only contains 3 distinct operations: $skip\_connect$, sep_conv_3x3 and avg_pool_3x3.

[3]Best genotype as published in [1]

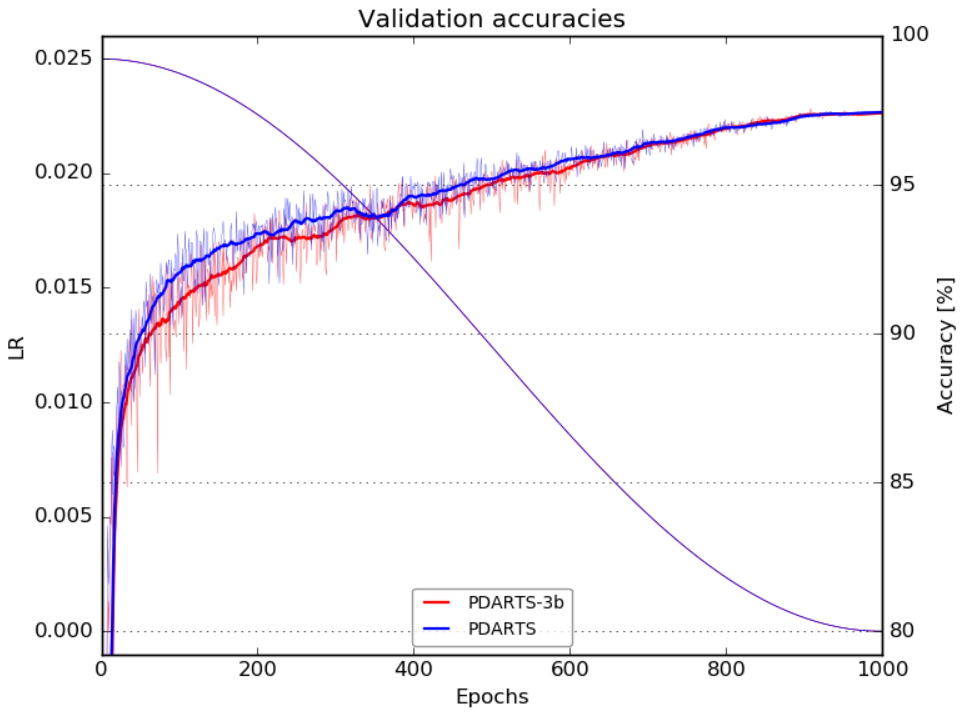


Figure 4.2: Evolution of validation accuracy of PDARTS and PDARTS-3b over 1000 epochs

4.1.2 Normal Blocks

We perform architecture search from scratch using the modified PDARTS implementation with standard settings to search for optimal cells of the types: normal1, normal2, normal3 and reduction. The results look quite optimistic. We achieve an increase of +1.74% on average in terms of validation accuracy with a batch size of 64, having the same search cost of 0.4 GPU-days as the original implementation (see Table A.2).

The relatively small initial network architecture of search-stage 1 could lead to problems for finding good architectures with the extended number of cell types. Because having a number of just 5 cells in stage 1, where 2 are of type reduction, we get a network where each of the normal cell types is present only once. This may lead to the exclusion of operations in the early architecture search that may possible be a good choice when stacking more cells. Thus, we perform another architecture search where we change the number of cells per stage from 5-11-17 to 11-14-17, what will increase the time necessary for search, of course, but may be a better estimate.

genotype	parameter size [M]	val accuracy [%]
PDARTS[4]	3.43	97.54
PDARTS	3.65 ± 0.05	97.43 ± 0.04
PDARTS-N3	2.2	97.17
PDARTS-N3-r	3.13	97.35
PDARTS-N3ex	2.56	97.28
PDARTS-N3ex-r	3.32	97.45

Table 4.2: Evaluation of architectures (r for restricted skip-connections according to [1], ex for extended search)

When we have a look at Table 4.2, we notice that the increased validation accuracy from the search phase vanished for final evaluation. By having a closer look, we observed something special in the found structure of the cells. While there are no skip connections in normal1 of PDARTS-N3, we do have 4 in normal2, and 7 in normal3 cell. Similar observations were made for PDARTS-N3ex, which performed better than PDARTS-N3 and achieved similar results to the original implementation when applying the restriction on skip-connections.

4.2 Search Space Analysis

We conduct experiments using DARTS and PDARTS, where we exclude each operation within the search space, one by one, to analyze their importance for the performance of searched architectures.

The exclusion of pooling operations ($3x3\_max\_pool$ and/or $3x3\_avg\_pool$) elevates the insertion of skip-connections to the architecture, what is known to be a bad option. Thus, the pooling operations are of importance for the performance of differentiable architecture searches. Of similar importance are the separable convolution operations, $3x3\_sep\_conv$ and $5x5\_sep\_conv$, as we observe worse validation accuracy in search phase of DARTS when excluding them from the search space. Only for the exclusion of $3x3\_dil\_conv$ and $5x5\_dil\_conv$ we can't notice any performance drop during architecture search in

[4] Best genotype as published in [1]

DARTS, nor in PDARTS. Thus, we evaluate the resulted genotypes in the final training. As a result, we get surprisingly not only comparable, but better validation accuracies for genotypes found with DARTS (see Table 4.3). With architectures found by PDARTS we can not observe an improvement.

genotype	parameter size [M]	val accuracy [%]
DARTS	2.80 ± 0.34	97.04 ± 0.10
DARTS-noDil	3.73 ± 0.15	$\mathbf{97.38} \pm 0.02$
PDARTS-noDil	3.75 ± 0.08	97.27 ± 0.20

Table 4.3: Evaluation of architectures with and without dilated convolutions (averaged over 2 runs)

4.2.1 Balanced Convolution Operations

In section 3.4.1 we noticed an imbalance of convolutional operations in terms of depth. In the following, we conducted experiments using PDARTS where we matched their depths. During search, we observe an average increase of validation accuracy of +0.92% and +0.32% for the two strategies proposed in section 3.4.1 (averaged over 2 runs). We name the resulting genotypes PDARTS-DD for 2 consecutive dilated convolution and PDARTS-DC for having a separable convolution appended to the dilated convolution.

The results are visible in Table 4.4. We can see that the genotypes of both strategies have a large number of parameters, which makes sense since we increased depth. PDARTS-DC achieves better results than PDARTS-DD when it comes to validation accuracy, while also containing less trainable parameters. Thus, appending a separable to the dilated convolution definitely is the better option than having 2 dilated convolutions in a row. Compared to the performance with the original operation, PDARTS-DC seems to achieve similar, if not better results.

genotype	parameter size [M]	val accuracy [%]
PDARTS[5]	3.43	97.54
PDARTS	3.65 ± 0.05	97.43 ± 0.04
PDARTS-DD	3.91	97.42
PDARTS-DC	3.76	97.51

Table 4.4: Evaluation of architectures with increased depth for dilated convolutions

The same experiments were executed using the original DARTS implementation. But all resulting genotypes ended up to contain not a single dilated convolution operation, what made final evaluation unnecessary.

4.2.2 New Operation

We add the new operation as introduced in section 3.4.2 to the search space and perform architecture searches using PDARTS. Resulted architectures did contain one or more edges with the new operation, but didn't yield any improvements in the final evaluation.

[5] Best genotype as published in [1]

4.3 Discretization by Grouping

In section 3.5, we introduced a grouping strategy that changes the selection policy at the end of the search phase. Following the proposed groups, we conducted experiments for first, the original implementation, and second, the architecture setup as proposed in section 3.3.2. We do this, because the results of section 4.1.2 looked promising in search phase, but somehow failed in producing good results after the discretization step.

We evaluate found genotypes for the original setup, once discretised with the classical approach and once with our grouping strategy, but observe worse results with genotypes following the grouping approach (see Table A.3 for results). Note that convergence for PDARTS-2g of Table A.3 is not ensured, which would explain the bad result. Similar is possible for PDARTS-1g, but less significant. See Figure A.3 for validation accuracy evolution in comparison.

In our second experiment series with additional cell types we observe interesting results. We again searched for genotypes the same way we did in section 3.3.2, for both, the classical setup and the one with extended stages. For each of the searches we evaluate the genotype found by the original selection procedure (including skip restriction) and the genotype found by the grouping selection, once with skip-connection restriction and once without.

In all cases, final evaluation of the genotypes following the grouping discretization result in better validation accuracy, as visible in Table 4.5. We even come close to the performance of the published genotype PDARTS of [1], while using less parameters by an amount of -15% and -25% for PDARTS-N3-1g-r and PDARTS-N3-2g, respectively.

genotype	parameter size [M]	batch size	lr	val accuracy [%]
PDARTS[6]	3.43	96	0.025	97.54
		64	0.015	97.45
PDARTS-N3-1-r	2.89	96	0.025	97.27
PDARTS-N3-1g-r	**2.94**			**97.46**
PDARTS-N3-1g	2.75			97.30
PDARTS-N3-2-r	3.40	64	0.015	97.29
PDARTS-N3-2g-r	3.26			97.30
PDARTS-N3-2g	**2.56**			**97.43**

Table 4.5: Evaluation of architectures with additional cell types and grouping strategy (g for grouping and r for restricted skips)

[6] Best genotype as published in [1]

Chapter 5

Discussion

With the restriction applied to the structure of cells by reducing the number of intermediate nodes, we could reduce the time necessary for search by approximately $\frac{1}{3}$ while achieving comparable results with increased number of layers to match parameter sizes. An extensive evaluation for one of the found genotypes even outperformed the best genotype as published in PDARTS [1]. These results are probably achieved by a better estimation during the optimization process because of a sparser set of optimization parameters. Supported by the fact that we have less variance in the results of different searches. One could argue that the depth of the network is responsible for the increase in performance. But since the our best genotype only has a in-cell-depth of 1, as visible in Figure 4.1, and a cell containing 4 intermediate node theoretically can have a in-cell-depth of 4, this is unlikely.

On the other hand, the added freedom by introducing 2 additional normal-cell types doesn't work out well. But it is still interesting to analyze how the cells built up. There are basically never any skip-connections present within cells of the first normal block, whereat they are very dominant in the later blocks. Also to mention that the first normal block never contained any dilated convolutions, therefore they must be of less relevance for the early feature extractions of the network. One reason of the approach to not increase the performance of architectures may be the same as above. Since we double the size of architecture weights, which we try to optimize over, the estimate of the final architecture is to noisy and thus results in bad performance after the discretization step. Supported by the fact that we get a better estimate when using a larger architecture for the search-process. A derived version of differentiable architecture search, such as freezing the structure and optimizing over just one cell type at the time could possibly help the approach to produce architectures where the different cell types harmonize better. Also assigning a different search space to each of the cell types could be considered.

While failing in improving the performance of the original setup, the grouping discretization strategy can improve the performance of architectures with 3 distinct normal cells. This may be the case because we have more noise in the estimation of architectures with a total of 4 distinct cells than we have with just 2. We can thus interpret our strategy to upper bound the estimation-error, reasoning with only having an influence for the 4-distinct-cells networks. Surprising results are achieved when combining the discretization strategy with a larger architecture used for search. Both have proven to mitigate the estimation-error and boost performance, and the combination seems to elevate this effect even more. We can see that by having a look at the results of Table 4.5, where we achieve the best result by not restricting the number of skip-connection as it was recommended in [1] or [5]. Meaning that the estimation of optimal architecture, using $1)$ a larger architecture and $2)$ the grouping discretization method, has already taken care of inserting an optimal number

of skip-connections. The result is quite remarkable since it contains only 2.56M parameters while achieving nearly the same validation accuracy as the best genotype of [1] which has 3.43M parameters, and also being close to the performance of the genotype published in [2], which contains 2.8M parameters. We need to mention here, that in terms of flops, our architecture performs 551M operations whereas the published genotype of PDARTS performs 543M[1]. So, the benefit restricts to the number of parameters and doesn't account for the number of operations as well. It has to be analyzed in pursuing experiments if no further restrictions are necessary when using this approach. Also, considering a better grouping concept of the operations may increase the performance of the approach. Analysis of the relations between operations within the architecture search process using an appropriate metric, e.g. the energy of operation or covariance of operation weights, could lead to better results.

The importance of dilated convolutions for well performing architectures has been questioned. While DARTS achieves better results without, PDARTS stays bellow its performance if excluded. We can again argue that the exclusion of operations, thus reduction of the optimization space leads to more reliable solution, especially with DARTS where we are anyway only slightly better than a randomly sampled architecture (as stated in [6]). The importance of dilated convolutions may thus be explained by the gap between the results of DARTS without them, and the results achieved by PDARTS where they're included. But as our well-performing genotype from Figure 4.1 shows no dilated convolutions within any cell, this question is not finally answered. Further does increasing the depth of dilated convolutions seem to lead to well performing architectures, but more experiments need to be conducted to get a better estimate, leading to a conclusion. We can also refer to the sharpDARTS paper [4], where a similar approach was pursued.

Last but not least can we say, that appending a deeper separable convolution operation to the search space doesn't help to improve in any aspect.

[1] calculated using https://github.com/Lyken17/pytorch-OpCounter

Appendix A

Appendix

A.1 PDARTS Algorithm

In A.1 we have a detailed overview of the proposed search algorithm of PDARTS [1]. In the illustration, we see an example setup for PDARTS, starting off with a small architecture (a) of 5 cells (3 normal- and 2 reduction cells), where we decide which operations seem the less promising for each edge and exclude them from the edge-specific search space. For the intermediate stage (b), we add 2 normal cells to each of the normal-blocks, ending up with a total of 11 cells and repeat the exclusion of operations per edge. In the final stage (c), we have again 2 more normal cells per block beside a sparse combination of operations per edge. After running the bi-level optimization, the final cell representation is determined out of the remaining operations.

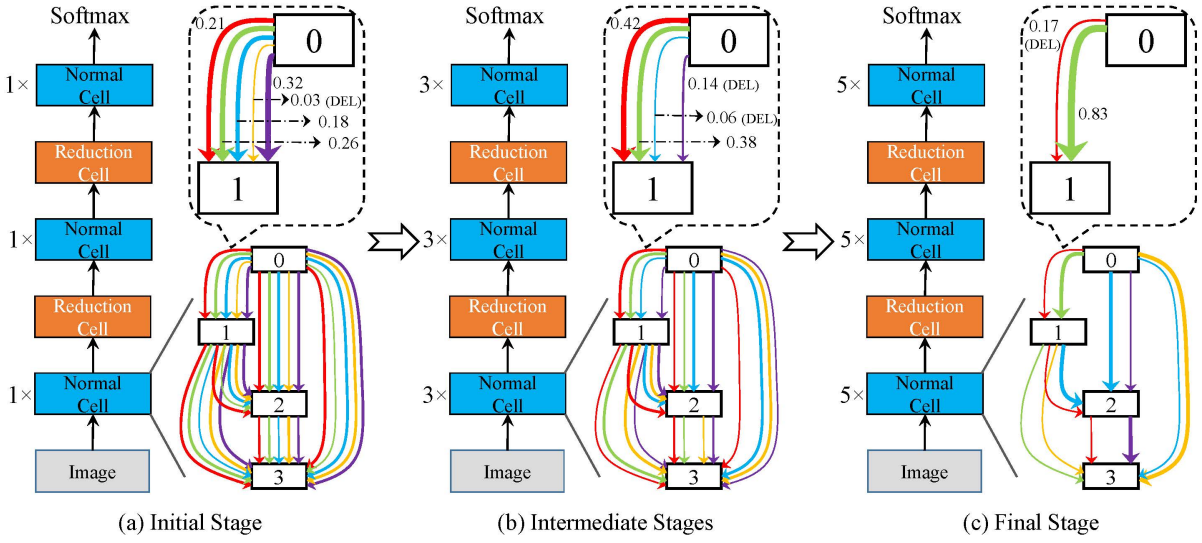


Figure A.1: Detailed view on staged architecture search of PDARTS [1]

A.2 Additional Results Material

nodes	batch size	validation accuracy	GPU-days
4	32	86.32 ± 0.26	0.7
	64	83.17 ± 0.18	**0.4**
3	32	85.63 ± 0.36	0.46
	64	83.47 ± 0.25	**0.27**

Table A.1: Architecture search with different number of intermediate nodes

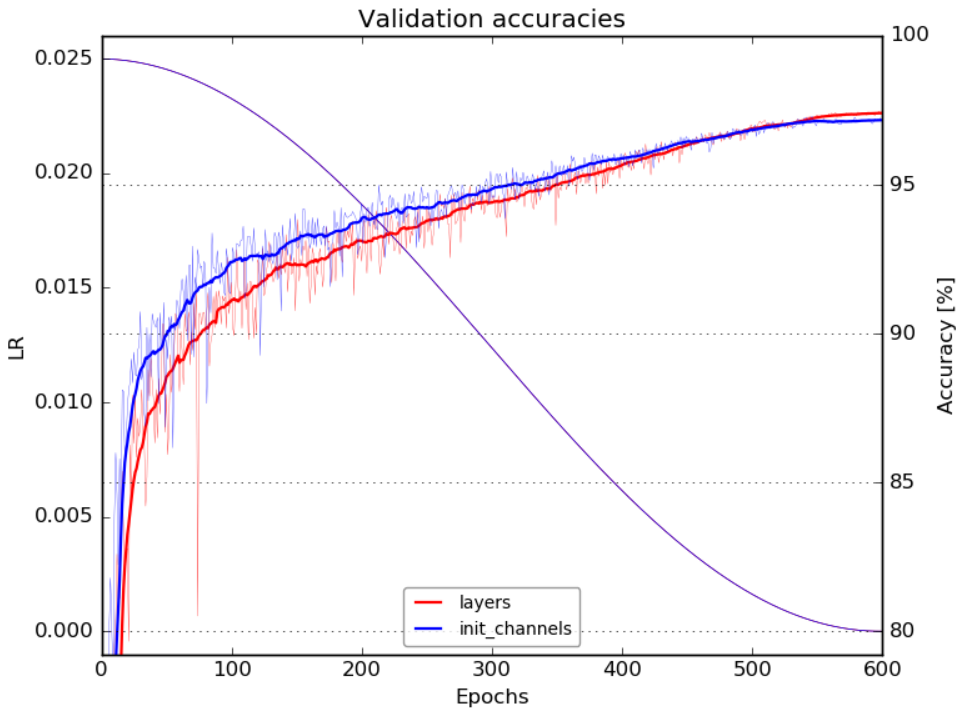


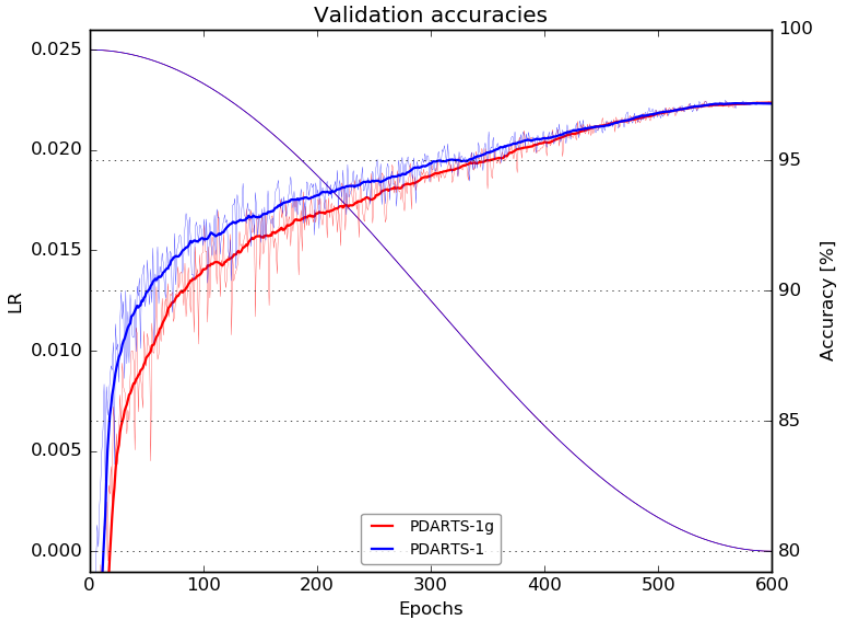
Figure A.2: Evolution of validation accuracy of PDARTS-3a with increased layers and initial channels

setup	batch size	validation accuracy	GPU-days
original	32	86.32 ± 0.26	0.7
	64	83.17 ± 0.18	0.4
3 normal	32	86.93 ± 0.31	0.64
	64	84.91 ± 0.23	0.4
3 normal (extended arch)	32	86.91 ± 0.24	0.92

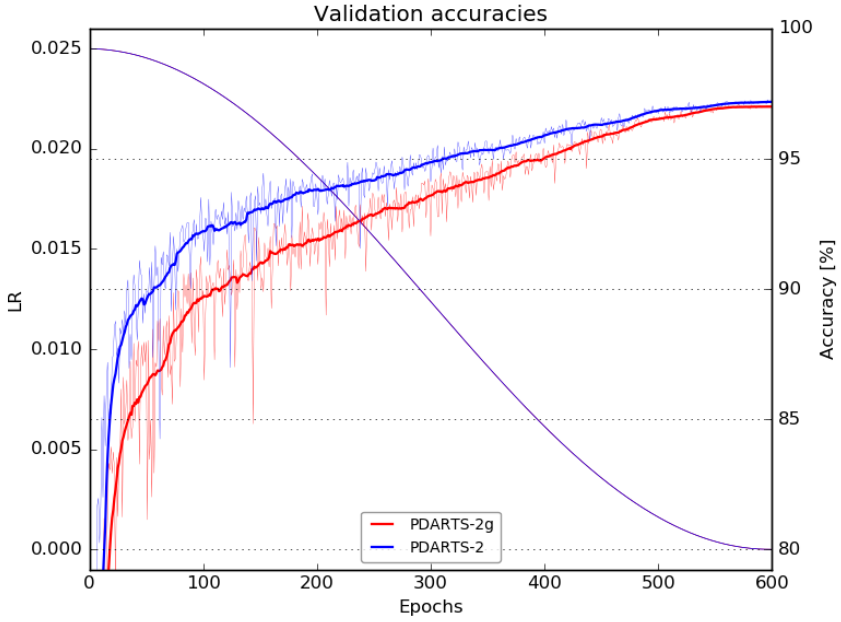
Table A.2: Architecture search with increased number of cell types

genotype	parameter size [M]	val accuracy [%]
PDARTS-1	3.54	97.30
PDARTS-1g	3.50	97.27
PDARTS-2	3.16	97.27
PDARTS-2g	3.77	97.09

Table A.3: Evaluation of architectures with grouping strategy (indicated by g)

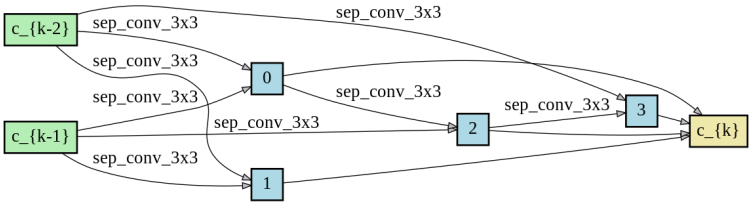


((a)) PDARTS-1g (red) vs. PDARTS-1 (blue)

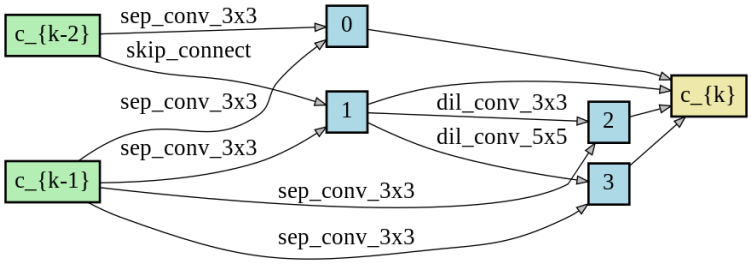


((b)) PDARTS-2g (red) vs. PDARTS-2 (blue)

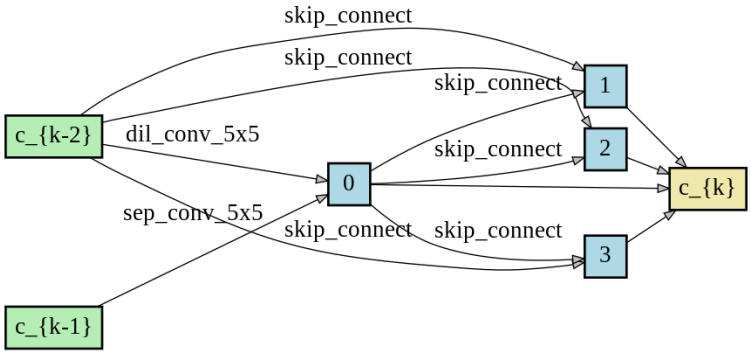
Figure A.3: Evolution of validation accuracy with and without operation grouping



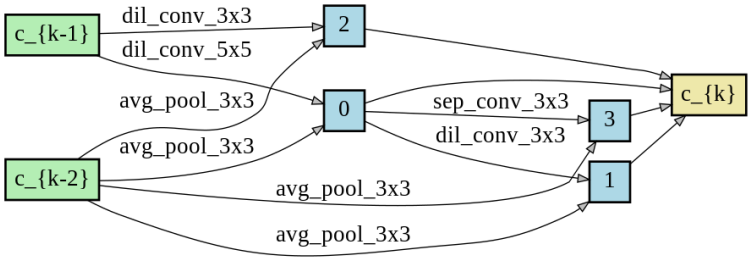
((a)) Normal cell



((b)) Normal2 cell



((c)) Normal3 cell



((d)) Reduction cell

Figure A.4: Genotype of PDARTS-N3-2g

Bibliography

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

[2] Xiangxiang Chu, Tianbao Zhou, Bo Zhang, and Jixiang Li. Fair darts: Eliminating unfair advantages in differentiable architecture search, 2019.

[3] Hyeong Gwon Hong, Pyunghwan Ahn, and Junmo Kim. Edas: Efficient and differentiable architecture search, 2019.

[4] Andrew Hundt, Varun Jain, and Gregory D. Hager. sharpdarts: Faster and more accurate differentiable architecture search, 2019.

[5] Hanwen Liang, Shifeng Zhang, Jiacheng Sun, Xingqiu He, Weiran Huang, Kechen Zhuang, and Zhenguo Li. Darts+: Improved differentiable architecture search with early stopping, 2019.

[6] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search, 2018.

[7] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. Pc-darts: Partial channel connections for memory-efficient differentiable architecture search, 2019.

[8] Zhao Zhong, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. Practical block-wise neural network architecture generation, 2017.