Single-DARTS: Towards Stable Architecture Search

Pengfei Hou*
houpengfei2020@126.com

Ying Jin*
Tsinghua University
sherryying003@gmail.com

Yukang Chen
The Chinese University of Hong Kong
yukangchen@cse.cuhk.edu.hk

Abstract

Differentiable architecture search (DARTS) marks a milestone in Neural Architecture Search (NAS), boasting simplicity and small search costs. However, DARTS still suffers from frequent performance collapse, which happens when some operations, such as skip connections, zeroes and poolings, dominate the architecture. In this paper, we are the first to point out that the phenomenon is attributed to ***bi-level*** *optimization.*

We propose ***Single-DARTS*** *which merely uses single-level optimization, updating network weights and architecture parameters simultaneously with the same data batch. Even single-level optimization has been previously attempted, no literature provides a systematic explanation on this essential point. Replacing the bi-level optimization, Single-DARTS obviously alleviates performance collapse as well as enhances the stability of architecture search. Experiment results show that Single-DARTS achieves state-of-the-art performance on mainstream search spaces. For instance, on NAS-Benchmark-201, the searched architectures are nearly optimal ones. We also validate that the single-level optimization framework is much more stable than the bi-level one. We hope that this simple yet effective method will give some insights on differential architecture search. The code is available at https://github.com/PencilAndBike/Single-DARTS.git.*

1. Introduction

Neural architecture search (NAS) has helped to find more excellent architectures than manual design. Generally, NAS is formulated as a bi-level optimization problem[2]:

$$\begin{aligned} \alpha^* &= \arg\min_{\alpha \in \boldsymbol{A}} \mathcal{L}_{val}(\alpha, w_\alpha^*) \\ s.t.\ w_\alpha^* &= \arg\min_{w_\alpha} \mathcal{L}_{train}(\alpha, w) \end{aligned} \tag{1}$$

where α denotes architecture, $\boldsymbol{A}$ denotes architecture search space, w_α denotes the network weights with the architecture parameter α, $\mathcal{L}_{train}$ and $\mathcal{L}_{val}$ denote the optimization loss on the training and validation dataset. Due to the inner optimization on $\mathcal{L}_{train}$ where each architecture should be well-trained respectively, it costs huge computational resources to search the optimal architecture. To avoid training each architecture from scratch, weight-sharing methods [27] are proposed, which constructs a super network where all architectures share the same weights. DARTS relaxes the search space to be continuous and approximates w_α^* by adapting w with only a single training step, instead of solving the inner optimization 1 completely. The approximation scheme is: $\nabla_\alpha \mathcal{L}_{val}(\alpha, w_\alpha^*) \approx \nabla_\alpha \mathcal{L}_{val}(\alpha, w - \xi \nabla_w \mathcal{L}_{train}(\alpha, w))$. It saves computations and finds competitive networks.

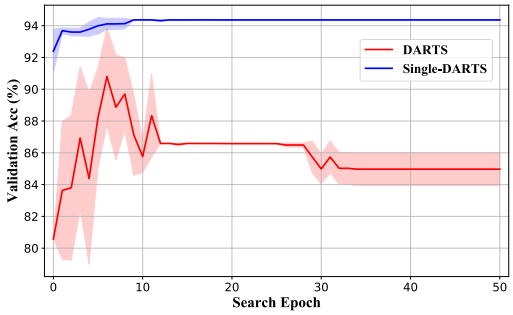


Figure 1: Search process comparison between the original DARTS and Single-DARTS.

However, many papers [34, 26, 37, 21, 12, 15] have reported the performance collapse in DARTS, where it easily converges to non-learnable operations, such as skip-connection, pooling, and zero, which devastates the accuracy of the searched model. As shown in Figure 1, at the early stage of searching, DARTS performs very unsteadily, and after one point when it converges to non-learnable operation, its performance drops. They also propose some effective mechanisms or regularization to improve DARTS.

In this paper, we dig deeper into this situation. First, we propose that the performance collapse in DARTS is **irreversible**. Moreover, we delve into the irreversible performance collapse in DARTS from the perspective of **optimization**, where little attention has been paid to in pre-

*Equal Contribution

vious works. In the bi-level optimization in DARTS, network weights and architecture parameters are updated alternatively with different batches of data (train and validation data). We derive that in such framework, learnable operations act as adding noise on input and fitting input worse than non-learnable operations like skip-connection and pooling. As a result, the probabilities of non-learnable operations increase faster than learnable ones, which enables non-learnable operations to dominate the search phase.

To this end, we propose **Single-DARTS**, a simple yet effective single-level optimization framework, which updates network weights and architecture parameters simultaneously with the identical batch of data. Through gradient analysis, we present that with such an optimization mechanism, Single-DARTS can obviously alleviate the performance collapse. Experiment results prove that Single-DARTS shows strong performance both in accuracy and stability. In Figure 1, Single-DARTS performs much more steadily than DARTS as well as reaches higher accuracy. On NAS-Benchmark-201 [15], Single-DARTS searches nearly the optimal architecture with a variance of 0, showing excellent stability.

In general, our contributions are as follows:

- We analyze the irreversible performance collapse in DARTS from the perspective of optimization. We provide theoretical insights by gradient analysis, by which we derive that it is the bi-level optimization framework in DARTS that causes severe and irreversible performance collapse.
- We propose Single-DARTS with a single-level optimization framework where the network weights and architecture parameters are updated simultaneously.
- Our simple yet effective method, Single-DARTS, reaches the state-of-the-art performance in mainstream search spaces. It also show strong stability.

2. Related Works

Neural Architecture Search. Neural architecture search (NAS) is an automatic method to design neural architecture instead of human design. Early NAS methods adopt reinforcement learning (RL) or evolutionary strategy [38, 2, 3, 31, 30, 39] to search among thousands of individually trained networks, which costs huge computation sources. Recent works focus on efficient weight-sharing methods, which falls into two categories: one-shot approaches [6, 4, 1, 7, 18, 33, 29] and gradient-based approaches [32, 27, 9, 8, 20, 12, 34, 23], achieve state-of-the-art results on a series of tasks [10, 17, 24, 35, 16, 28] in various search spaces. They construct a super network/graph which shares weights with all sub-network/graphs. The former commonly does heuristic search and evaluates sampled architectures in the super network to get the optimal architecture. The latter relaxes the search space to be continuous and introduces differential and learnable architecture parameters. In this paper, we mainly discuss gradient-based approaches.

Differentiable Architecture Search. Gradient-based approaches are commonly formulated as an approximation of the bi-level optimization which updates network weights and updates architecture parameters in the training and validation dataset alternatively[27]. It has searched competitive architectures, however, many papers pointed out DARTS-based methods don't work stably, suffering from severe performance collapse. NAS-Benchmark-201 [15] points out that DARTS performs badly in this search space. Its performance drops quickly during the search procedure. Towards this serious problem, researchers try to give explanations and solve it. [4] shows the relationship between DARTS searched architectures' performance and the domain eigenvalue of $\nabla^2_\alpha \mathcal{L}_{valid}$. FairDARTS [12] shows that DARTS is easy to converge to skip-connect operation due to its unfair advantage in exclusive competition. It gives each operation an independent weight to replace the exclusive competition and introduces zero-one loss to decrease the unfair advantage. However, in our point bi-level optimization is the key reason for the collapse and single-level optimization could also get satisfied result under exclusive competition. Previous works[26, 34, 19, 21] show that DARTS favors non-parametric operations. DropNAS [19] observes the co-adaption problem and Matthew effect that light operations are trained maturely earlier. Different from these works, we analyze DARTS from the perspective of optimization.

Single-level Optimization. Although the original paper [27] shows that single-level optimization performs worse than bi-level optimization and they indicate that single-level would cause overfitting, but recent works show the opposite results [21, 5, 19]. They adopt single/one-level optimization to replace bi-level optimization in their methods. Combining single-level optimization with their proposed methods, they can find more accurate network architectures. StacNAS[21] advocates that the over-fitting phenomenon is caused by the network's depth gap between the search phase and the evaluation phase. GoldNAS[5] indicates that super-network parameters are trained much more effectively than architecture parameters and adds data augmentation in the search phase. DropNAS [19] combines operations dropout with single-level optimization to solve the co-adaptation problem it proposes. However, on one side, though based on single-level optimization, these works focus on other techniques to get satisfactory results. Our method just uses single-level optimization alone to search steadily high-performance architectures. On the other side, these works prove that single-level optimization outper-

forms bi-level optimization empirically, without detailed explanations or proof. On the contrary, in this paper, we give theoretical insights about bi-level optimization from the perspective of optimization, on the basis of which we propose a single-level method, Single-DARTS.

3. Irreversible Collapse in DARTS

In this section, we first review the Differentiable Architecture Search (DARTS) framework. Then we propose that the performance collapse in DARTS is irreversible.

Preliminary of DARTS Differentiable Architecture Search (DARTS) is a milestone in neural architecture search. Its search space includes stacks of several cells, where each cell is a directed acyclic graph. Each cell consists of sequential nodes where node i represents the latent feature map x^i. The edge from node i to node j represents a connection, and each connection is one operation from candidate operations $\mathbb{O}$: convolution, pooling, zero, skip-connect, etc. Let $o^{i,j}$ denote the operation from node i to j. The output of each intermediate node is computed based on all of its predecessors: $x_j = \sum_{i<j} o^{i,j}(x_i)$ In DARTS, it transforms the problem of searching the best architecture to searching the best operation on each connection. DARTS makes the search space continuous by relaxing the categorical choice of a particular operation to a weighted sum over all possible operations. Let $\alpha_k^{i,j}$ denote architecture parameter with operation o_k between node i and node j, then the output of this connection is

$$\bar{o}_{i,j}(x_i) = \sum_k \frac{exp(\alpha_k^{i,j})}{\sum_{k'} exp(\alpha_{k'}^{i,j})} o_k(x_i) \tag{2}$$

where it adopts softmax function to have weights over different operations. The output of one intermediate node is computed based on all of its predecessors: $x_j = \sum_{i<j} \bar{o}_{i,j}(x_i)$

Therefore, DARTS needs to optimize the network weights w and the architecture parameters α. DARTS resolves this problem by a bi-level optimization framework, which updates network weights on the train set and architecture parameters on the validation set: $\nabla_\alpha \mathcal{L}_{val}(\alpha, w_\alpha^*) \approx \nabla_\alpha \mathcal{L}_{val}(\alpha, w - \xi \nabla_w \mathcal{L}_{train}(\alpha, w))$.

Irreversible Performance Collapse Previous works have unveiled that it is easy for DARTS to converge to non-learnable operations like skip-connect, zero, pooling, etc. As a result, the performance collapse will happen. In this paper, we delve deeper into this situation. As shown in Appendix Figure A-1, we train DARTS in the two spaces and plot the probability curves of one edge. At one point, the probability of non-learnable operation starts to surpass the learnable one. However, for this edge, the ideal choices are actually learnable operations, and non-learnable operations will bring about disastrous influence to model accuracy. Thus, the divergence between them becomes increasingly larger, enabling non-learnable operations to dominate the architecture search. Such divergence also indicates that the performance collapse is irreversible.

4. Methodology

In this section, we firstly understand DARTS from the perspective of information gain. Then we theoretically analyze the bi-level optimization and explain the collapse in DARTS. At last, we present our solution, Single-DARTS.

4.1. Understanding DARTS

According to Eq.2, we consider a general formulation for the connection setting in DARTS:

$$f(\sum_i p_i x_i), where \sum p_i = 1, 0 \leq p_i \leq 1$$

where x_i is a set of input vectors and its corresponding loss function about one-hot vector y is:

$$\mathcal{L} = -y^T \ln(\text{softmax}(f(\sum_i p_i x_i))) \tag{3}$$

We denote $\bar{x}$ as $\sum p_i x_i$. If model fits y with x_i better than x_j, it means the cross entropy between y and $f(x_i;\theta)$ is smaller than $f(x_j;\theta)$. For loss function 3, p_i should be increased faster than p_j for x_i gets more information gain than x_j. Under gradient descend methods, the gradient of p_i is smaller than p_j. Heuristically, $\frac{\partial \mathcal{L}}{\partial \bar{x}}^T(x_i)$ *reflect the advantage of input* x_i. We give the formal description as follows: for one hot random vector y, function $f(x;\theta)$ and loss $\mathcal{L} = -y^T \ln(f(\sum p_k x_k;\theta)$, if $-y^T \ln(f(x_i;\theta)) \leq -y^T \ln(f(x_j;\theta))$, then we have $\frac{\partial \mathcal{L}}{\partial p_i} \leq \frac{\partial \mathcal{L}}{\partial p_j}$. Limited to mathematical tools, we consider a simplified situation where $f(x;\theta) = x$ to give some insight and we conduct empirical experiments to verify the following theorems in deep neural networks.

Theorem 4.1 *For input vectors* $\{x_k\}$, $\mathcal{L} = -y^T \ln(\text{softmax}(\sum_k p_k x_k))$ *where* y *is one-hot vector,* $\sum_k p_k = 1$ *and* $0 \leq p_k \leq 1$, $\bar{x}$ *as* $\sum_k p_k x_k$, *if KL-Divergence*$(p(\bar{x})||p(x_k)) \approx 0 \; \forall \; x_k$ *and*

$$-y^T \ln(\text{softmax}(x_i)) \leq -y^T \ln(\text{softmax}(x_j)) \tag{4}$$

then

$$\frac{\partial \mathcal{L}}{\partial p_i} \lessapprox \frac{\partial \mathcal{L}}{\partial p_j} \tag{5}$$

Proof 4.1 *We have* $\frac{\partial \mathcal{L}}{\partial p_i} = (\text{softmax}(x_i) - y)^T x_i$, *let* t *denote* $\text{softmax}(\bar{x})$. *Suppose* $y^0 = 1$ *and* $y^l = 0$ *where* $l > 0$, *we have*

$$-y^T \ln \text{softmax}(x_i) = -\ln \frac{\exp x_i^0}{\sum_l \exp x_i^l}$$

We have

$$-\ln \exp x_i^0 + \ln \sum_l \exp x_i^l \leq -\ln \exp x_j^0 + \ln \sum_l \exp x_j^l$$

$$x_i^0 - x_j^0 \geq \ln \frac{\sum_l \exp x_i^l}{\sum_l \exp x_j^l}$$

And

$$\frac{\partial \mathcal{L}}{\partial p_i} = (\text{softmax}(\bar{x}) - y)^T x_i = \sum_l t^l x_i^l - x_i^0$$

$$\frac{\partial \mathcal{L}}{\partial p_i} - \frac{\partial \mathcal{L}}{\partial p_j} = (\sum_l t^l x_i^l - x_i^0) - (\sum_l t^l x_j^l - x_j^0)$$

$$= \sum_l t^l (x_i^l - x_j^l) - (x_i^0 - x_j^0)$$

To prove $\frac{\partial \mathcal{L}}{\partial p_i} \leq \frac{\partial \mathcal{L}}{\partial p_j}$, *we just need to prove* $\sum_l t^l (x_i^l - x_j^l) \leq \ln \frac{\sum_l \exp x_i^l}{\sum_l \exp x_j^l}$, *then we have* $\frac{\partial \mathcal{L}}{\partial p_i} - \frac{\partial \mathcal{L}}{\partial p_j} \leq \ln \frac{\sum_l \exp x_i^l}{\sum_l \exp x_j^l} - (x_i^0 - x_j^0) \leq 0$.
Let q_j *denote* $\text{softmax}(x_j)$, *according to Jensen's inequality that for convex function* $h(x)$, $h(\sum_k p_k x_k) \leq \sum h_k f(x_k)$ *and* $-\ln(x)$ *is a convex function, we have*

$$\ln \frac{\sum_l \exp x_i^l}{\sum_l \exp x_j^l} = \ln \sum_l \frac{\exp x_j^l}{\sum_{l'} \exp x_j^{l'}} \exp(x_i^l - x_j^l)$$

$$\geq \sum_l q_j^l \ln(\exp(x_i^l - x_j^l))$$

$$= \sum_l q_j^l (x_i^l - x_j^l)$$

Since KL-Divergence$(p(\bar{x})||p(x_k)) \approx 0$, *thus* $\text{softmax}(\bar{x}) \approx \text{softmax}(x_k)$, $\sum_l q_j^l (x_i^l - x_j^l) \approx \sum_l t^l (x_i^l - x_j^l)$. *So* $\sum_l t^l (x_i^l - x_j^l) \leq \ln \frac{\sum_l \exp x_i^l}{\sum_l \exp x_j^l}$.

Note that in Theorem 4.1 we assume KL-Divergence$(p(\bar{x})||p_k(x)) \approx 0$. In fact, with the help of Batch Normalization (BN) in neural networks, the distribution of intermediate batch normalized feature maps can be approximated as normal Gaussian distribution. Moreover, $\bar{x}$ is the weighted sum of x_i so it can be approximated as normal Gaussian distribution as well. Therefore, our assumption is valid.

4.2. Drawback in Bi-level Framework

Consider a more concrete formulation that x_i are transformed features from the same input x, $x_i = o_i(x)$. Specifically, $o_i(x)$ could be linearly expanded as $o_i(x) = W_i x$.

$$\frac{\partial \mathcal{L}}{\partial p_i} = \mathbb{E}[\frac{\partial \mathcal{L}}{\partial \bar{x}}^T (W_i x); X, y] \tag{6}$$

$$\frac{\partial \mathcal{L}}{\partial W_i} = \mathbb{E}[p_i \frac{\partial \mathcal{L}}{\partial \bar{x}} x^T; X, y] \tag{7}$$

To be emphasized, Eq.7 only exist for learnable operations for non-learnbale operations **do not** update their weights. According to last section, if $W_i x$ bring more information gain than $W_j x$, then $\frac{\partial \mathcal{L}}{\partial p_i}$ should be smaller than $\frac{\partial \mathcal{L}}{\partial p_j}$. For DARTS, p_i is updated on the validation dataset. On the iteration t and on the validation dataset $\mathcal{D}_{\text{valid}} = \{X_{val}, y_{val}\}$:

$$\frac{\partial \mathcal{L}}{\partial p_i^t} = \frac{1}{N} \sum_{j=1}^{N} \frac{\partial \mathcal{L}(x_{val}^j)}{\partial \bar{x}}^T (W_i^t x_{val}^j)$$

W_i^t is updated on the training dataset $\mathcal{D}_{\text{train}} = \{X_{train}, y_{train}\}$:

$$\frac{\partial \mathcal{L}}{\partial W_i^t} = \frac{p_i}{M} \sum_{k=1}^{M} \frac{\partial \mathcal{L}(x_{train}^k)}{\partial \bar{x}} {x_{train}^k}^T$$

$$W_i^t = W_i^{t-1} - \eta \frac{\partial \mathcal{L}}{\partial W_i^{t-1}}$$

$$= W_i^{t-1} - \eta \frac{p_i^{t-1}}{M} \sum_{k=1}^{N} \frac{\partial \mathcal{L}(x_{train}^k)}{\partial \bar{x}} {x_{train}^k}^T$$

As a result, the gradient of p_i is:

$$\frac{\partial \mathcal{L}}{\partial p_i^t} = \frac{1}{N} \sum_{j=1}^{N} \frac{\partial \mathcal{L}(x_{val}^j)}{\partial \bar{x}}^T (W_i^{t-1} x_{val}^j)$$

$$- \frac{\eta p_i^{t-1}}{NM} \sum_{j=1}^{N} \sum_{k=1}^{M} (\frac{\partial \mathcal{L}(x_{val}^j)}{\partial \bar{x}}^T \frac{\partial \mathcal{L}(x_{train}^k)}{\partial \bar{x}}) ({x_{train}^k}^T x_{val}^j)$$

For bi-level optimization W and α are computed on the different batches, since samples are different, $\frac{\partial \mathcal{L}(x_{val}^j)}{\partial \bar{x}}$ is independent of $\frac{\partial \mathcal{L}(x_{train}^k)}{\partial \bar{x}}$, and x_{train}^k is independent of x_{val}^j. Thus

$$\sum_{j=1}^{N} \sum_{k=1}^{M} (\frac{\partial \mathcal{L}(x_{val}^j)}{\partial \bar{x}}^T \frac{\partial \mathcal{L}(x_{train}^k)}{\partial \bar{x}}) ({x_{train}^k}^T x_{val}^j) \approx 0 \tag{8}$$

Furthermore, during the early stage of architecture search

$$\frac{\partial \mathcal{L}}{\partial p_i^t} \approx \frac{1}{N}\sum_{j=1}^{N} \frac{\partial \mathcal{L}(x_{val}^j)}{\partial \bar{x}}^T (W_i^{t-1} x_{val}^j) \tag{9}$$

$$= \frac{1}{N}\sum_{j=1}^{N} \frac{\partial \mathcal{L}(x_{val}^j)}{\partial \bar{x}}^T (W_i^{t-2} x_{val}^j) \tag{10}$$

$$- \frac{\eta p_i^{t-2}}{NM}\sum_{j=1}^{N}\sum_{k=1}^{M} (\frac{\partial \mathcal{L}(x_{val}^j)}{\partial \bar{x}}^T \frac{\partial \mathcal{L}(x_{train}^k)}{\partial \bar{x}})(x_{train}^k{}^T x_{val}^j) \tag{11}$$

$$\approx \frac{1}{N}\sum_{j=1}^{N} \frac{\partial \mathcal{L}(x_{val}^j)}{\partial \bar{x}}^T (W_i^{t-2} x_{val}^j) \tag{12}$$

$$\approx .... \approx \frac{1}{N}\sum_{j=1}^{N} \frac{\partial \mathcal{L}(x_{val}^j)}{\partial \bar{x}}^T (W_i^{0} x_{val}^j) \tag{13}$$

$$\tag{14}$$

Let i denote learnable operaions' indicator while j as the non-learnable's. Learnable operations (convolution) are initialized randomly thus they are actually adding noise on input x. So during the early stage, they bring less information gain than non-learnable operations. 9 does not establish when iterations are adequate, however, due to softmax as activation function, the gap between $\frac{\partial \mathcal{L}}{\partial p_j^t}$ and $\frac{\partial \mathcal{L}}{\partial p_i^t}$ could be expanded during early stage.

Effect of activation function For α_i and α_j, if $\alpha_j \geq \alpha_i$, the gradient of α_i is more likely to be smaller than α_j under softmax activation and gradient descent. It means the gap between α_i and α_j would be expanded. Formally, we have

Theorem 4.2 *For function* $\mathcal{L} = g(\sum_i \frac{exp(\alpha_i)}{\sum_j exp(\alpha_j)} x_i)$, *let* $p_i = \frac{exp(\alpha_i)}{\sum_j exp(\alpha_j)}$, $\bar{x} = \sum_i p_i x_i$, *if* $\frac{\partial \mathcal{L}}{\partial \bar{x}}^T x_j < \frac{\partial \mathcal{L}}{\partial \bar{x}}^T x_i$ *and* $\alpha_j \geq \alpha_i$, *then* $\frac{\partial \mathcal{L}}{\partial \alpha^j} \leq \frac{\partial \mathcal{L}}{\partial \alpha_i}$

The proof is included in Appendix. On the early stage, $\frac{\partial \mathcal{L}}{\partial \bar{x}}^T x_j \leq \frac{\partial \mathcal{L}}{\partial \bar{x}^T} x_i$ for learnable operation i and non-learnable operation j. Theorem 4.2 indicates that in DARTS, on the early training stage, once one non-learnable operation is superior to others, the probability of this operation will become increasingly larger, and after some iterations, it will be nearly 1, which leads to the irreversible performance collapse. Moreover, we could get the convergence speed as follows.

Theorem 4.3 *Given* n *operations, learning rate* η, $i^* = \arg\min_i \frac{\partial \mathcal{L}}{\partial \bar{x}}^T x_i$, *we define the margin between operations as* $\delta = \min_{i \neq i^*} \frac{\partial \mathcal{L}}{\partial \bar{x}}^T (x_i - x_{i^*}) \geq 0$, *if* $\alpha_{i^*} \geq \alpha_i$, *then* $\forall$ $\epsilon > 0$, *it achieves* $p_{i^*} > 1 - \epsilon$ *under gradient descent in iterations*

$$t \leq \frac{n \ln((1-\epsilon)n)}{\eta \delta} \tag{15}$$

The proof is included in Appendix. It shows that fewer candidate choices, a bigger margin and larger learning rates would cause performance collapse earlier. It is consistent with the empirical results that when the learning rate is lower, DARTS will perform better on NAS-Benchmark-201 if the total epoch is fixed to 50.

4.3. Solution: Single-DARTS

The analysis above indicates that it is the bi-level optimization in DARTS that plays a crucial role in performance collapse. In this paper, we propose Single-DARTS, which optimizes the network weights and architecture parameters by the same batch of data.

$$\alpha^t, w^t - = \eta \nabla_{\alpha, w} \mathcal{L}_{train}(\alpha^{t-1}, w^{t-1}) \tag{16}$$

Under this framework, we have

$$\frac{\partial \mathcal{L}}{\partial p_i^t} = \frac{1}{N}\sum_{j=1}^{N} \frac{\partial \mathcal{L}(x_{val}^j)}{\partial \bar{x}}^T (W_i^{t-1} x_{val}^j)$$

$$- \frac{\eta p_i^{t-1}}{N^2}\sum_{j=1}^{N} (\frac{\partial \mathcal{L}(x_{train}^j)}{\partial \bar{x}}^T \frac{\partial \mathcal{L}(x_{train}^j)}{\partial \bar{x}})(x_{train}^j{}^T x_{train}^j)$$

$$< \frac{1}{N}\sum_{j=1}^{N} \frac{\partial \mathcal{L}(x_{val}^j)}{\partial \bar{x}}^T (W_i^{t-1} x_{val}^j)$$

$$< ... < \frac{1}{N}\sum_{j=1}^{N} \frac{\partial \mathcal{L}(x_{val}^j)}{\partial \bar{x}}^T (W_i^{0} x_{val}^j)$$

It means for learnable operations, $\frac{\partial \mathcal{L}}{\partial p_i}$ in the single-level framework is smaller than in bi-level framework at the same iteration and $\frac{\partial \mathcal{L}}{\partial p_i}$ is decreased as the training process goes. After several iterations, learnable operations (convolution) are really learned but not add noise on the input, Thus they bring more information gain than non-learnable operations. Therefore, $\frac{\partial \mathcal{L}}{\partial p_i}$ of learnable operations can possibly be smaller than non-learnable operations. Under gradient descent, p_i would consequently be increased faster than non-learnable operations and finally exceed them.

Furthermore, we use sigmoid to replace softmax as activation function on α. For sigmoid, $\frac{\partial \mathcal{L}}{\partial \alpha_i} = p_i(1-p_i)\frac{\partial \mathcal{L}}{\partial \bar{x}}^T x_i$. $p_i(1-p_i)$ is a nonmonotone function and it could be nearly zero when p_i is near 1, thus even $p_i \geq p_j$ and $\frac{\partial \mathcal{L}}{\partial \bar{x}}^T x_i \leq \frac{\partial \mathcal{L}}{\partial \bar{x}}^T x_j$, it's possible that $\frac{\partial \mathcal{L}}{\partial \alpha_i} \geq \frac{\partial \mathcal{L}}{\partial \alpha_j}$. Therefore, from theoretical analysis, Single-DARTS can alleviate the irreversible performance collapse in DARTS.

5. Experiments

In this section, we conduct experiments to validate our theoretical insights and evaluate the performance of our proposed method, Single-DARTS. We choose two mainstream search spaces, NAS-Benchmark-201 and DARTS space.

5.1. Results

5.1.1 NAS-Benchmark-201

In NAS-Benchmark-201, we train the supermodel for 50 epochs with batch size of 64. We set SGD optimizer with learning rate as 0.005, weight decay as 3×10^{-4}, momentum as 0.9 and the cosine scheduler for network weights. And we set Adam optimizer with a fixed learning rate of 3×10^{-4} and a momentum of (0.5, 0.999) for architecture parameters. We run each method 5 times and report the average results.

Table 1 shows the experiment results on NAS-Benchmark-201. NAS-Benchmark-201 is a challenging search space, where DARTS performs poorly, showing very low accuracy consistently. On the contrary, by replacing the bi-level optimization with single-level optimization, Single-DARTS outperforms all the previous methods. Single-DARTS shows very strong stability and its results are very close to the optimal results. So Single-DARTS is superior to the previous method both in accuracy and stability.

5.1.2 DARTS

We directly do the search process on the ImageNet-1k dataset. Specifically, input images are downsampled third times by convolution layers at the beginning of the supermodel to reduce spatial resolution which follows [25]. In the search phase, we train the supermodel for 50 epochs. We set batch size as 360, learning rate as 0.025, weight decay as 3×10^{-4}, and the cosine scheduler for network weights. And we use Adam with a fixed learning rate of 3×10^{-4} and a momentum of (0.5, 0.999) for architecture parameters, and weight decay is set to 0 to avoid extra gradients on zero operations. For sigmoid, α is initialized as $-\ln(7)$ such that $sigmoid(\alpha) = 0.125$ (from our empirical experiments setting $\alpha = 0$ could be worse than using softmax, more comparisons are shown in 5 and Appendix). The search phase costs 4 GPUs for about 28 hours on NVIDIA GeForce RTX 2080ti. In the retraining phase, we adopt the training strategy as previous works [20] to train the searched architecture from scratch, without any additional module. The whole process lasts 250 epochs, using SGD optimizer with a momentum of 0.9, a weight decay of 3×10^{-5}. It's to be mentioned that due to memory limit of 2080ti, we scale both batch size and learning rate by 0.75 (1024 to 768, 0.5 to 0.375). And we reproduce previous methods [20, 34, 36] under this setting to ensure fairness. Additional enhancements are adopted including label smoothing and an auxiliary loss tower during training as in PDARTS. Learning rate warm-up is applied for the first 5 epochs and decayed down to zero linearly. The retrain phase costs 8 GPUs for about 3.5 days on RTX 2080ti.

As shown in Table 2, Single-DARTS search the state-of-the-art architecture with 77.0% top1 accuracy on ImageNet-1K. There is no constraint on FLOPs during the search process and larger models naturally have better performance. We report the result of previous methods in their original paper. When training the searched architectures from scratch, we follow the strategy of PDARTS, which re-trains the searched architecture for 250 epochs. In contrast, DropNAS trains for 600 epochs and DARTS+ for 800 epochs. AutoAugment, mixup and SE module are also applied in DropNAS.

We transfer the searched architecture to CIFAR10. To keep mobile setting, we shorten the number of layers from 20 to 14 to make parameters under 3.5M. The training setting is identical to PDARTS. The architecture's initial channels are 36. The whole process lasts 600 epochs with batch size of 128, using SGD optimizer with a momentum of 0.9, a weight decay of 3×10^{-4}. Cutout regularization of length 16, drop-path of probability 0.3 and auxiliary towers of weight 0.4 are applied. The initial learning rate is 0.025, which is decayed to 0 following the cosine rule. As shown in Table 3, Single-DARTS search the comparable result with 97.54% accuracy on CIFAR10. It's to be mentioned that DARTS+ trains for 2,000 epochs.

5.2. Analysis

5.2.1 Gradient Correlation

We define $\sum_{j=1}^{N}\sum_{k=1}^{M}(\frac{\partial \mathcal{L}(x_{val}^{j})}{\partial \bar{x}}^{T}\frac{\partial \mathcal{L}(x_{train}^{k})}{\partial \bar{x}})(x_{train}^{k}{}^{T}x_{val}^{j})$ as the gradient correlation in bi-level optimization, accordingly the one in single-level optimization is $\sum_{j=1}^{N}\sum_{k=1}^{M}(\frac{\partial \mathcal{L}(x_{train}^{j})}{\partial \bar{x}}^{T}\frac{\partial \mathcal{L}(x_{train}^{k})}{\partial \bar{x}})(x_{train}^{k}{}^{T}x_{train}^{j})$. In our analysis, we point out that it is the gradient correlation that plays a crucial role in the performance collapse in DARTS. Here we visualize the gradient correlation in bi-level optimization and single-level optimization. Figure 2 compares the gradients correlation 8 in DARTS and Single-DARTS. For DARTS, the gradients correlation is nearly zero. On the contrary, the gradient correlation in Single-DARTS is much bigger than the one in DARTS, which is consistent with our theoretical analysis.

5.2.2 Operator Gradients

We compare $\frac{\partial \mathcal{L}}{\partial p_i}$ in the last cell between DARTS and Single-DARTS in Figure 3. For DARTS, $\frac{\partial \mathcal{L}}{\partial p_i}$ of non-learnable operations are smaller than learnable operations and the gap increases as the training process goes. Thus

Table 1: Comparison of different NAS algorithms on NAS-Bench-201.

Method	CIFAR-10		CIFAR-100		ImageNet-16-120	
	validation	test	validation	test	validation	test
Optimal	91.61	94.37	73.49	73.51	46.77	47.31
Å RSPS[22]	$84.16_{\pm 1.69}$	$87.66_{\pm 1.69}$	$59.00_{\pm 4.60}$	$58.33_{\pm 4.34}$	$31.56_{\pm 3.28}$	$31.14_{\pm 3.88}$
DARTS[27]	$39.77_{\pm 0}$	$54.30_{\pm 0}$	$15.03_{\pm 0}$	$15.61_{\pm 0}$	$16.43_{\pm 0}$	$16.32_{\pm 0}$
GDAS[14]	$90.00_{\pm 0.21}$	$93.51_{\pm 0.13}$	$71.14_{\pm 0.27}$	$70.61_{\pm 0.26}$	$41.70_{\pm 1.26}$	$41.84_{\pm 0.90}$
SETN [13]	$82.25_{\pm 5.17}$	$86.19_{\pm 4.63}$	$56.86_{\pm 7.59}$	$56.87_{\pm 7.77}$	$32.54_{\pm 3.63}$	$31.90_{\pm 4.07}$
ENAS [29]	$39.77_{\pm 0}$	$54.30_{\pm 0}$	$15.03_{\pm 0}$	$15.61_{\pm 0}$	$16.43_{\pm 0}$	$16.32_{\pm 0}$
CDARTS [36]	$91.13_{\pm 0.44}$	$94.02_{\pm 0.31}$	$72.12_{\pm 1.23}$	$71.92_{\pm 1.30}$	$45.09_{\pm 0.61}$	$45.51_{\pm 0.72}$
DARTS- [11]	$91.03_{\pm 0.44}$	$93.80_{\pm 0.40}$	$71.36_{\pm 1.51}$	$71.53_{\pm 1.51}$	$44.87_{\pm 1.46}$	$45.12_{\pm 0.82}$
Single-DARTS	$\mathbf{91.55}_{\pm 0}$	$\mathbf{94.36}_{\pm 0}$	$\mathbf{73.49}_{\pm 0}$	$\mathbf{73.51}_{\pm 0}$	$\mathbf{46.37}_{\pm 0}$	$\mathbf{46.34}_{\pm 0}$

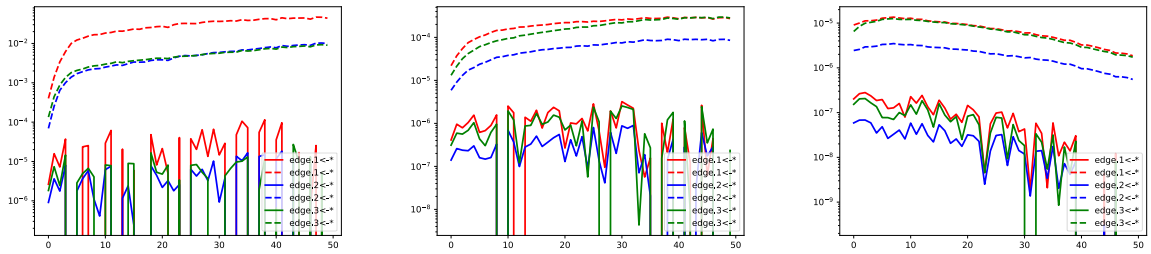
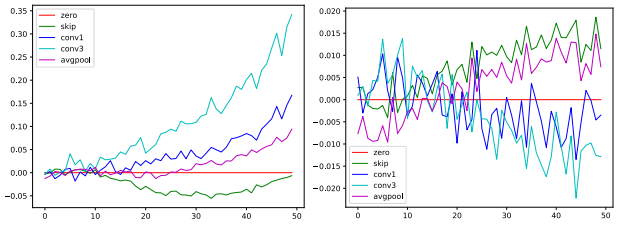


Figure 2: Comparison of gradient correlation 8 between bi-level and single-level optimization on CIFAR10 dataset in NAS-Benchmark-201 search space. It shows the gradient correlation on different nodes of the first, the middle, and the last cells. Solid lines represent bi-level and dashed lines represent single-level. Gradient correlation in bi-level optimization is nearly zero and far smaller than in single-level optimization.



(a) DARTS (b) Single-DARTS

Figure 3: Comparison of gradients of p_i in the edges 3←2 in the 15th cell in DARTS and Single-DARTS.

non-learnable operations' architecture parameters would be much larger than the learnable ones. On the contrary, in Single-DARTS, $\frac{\partial \mathcal{L}}{\partial p_i}$ of learnable operations compete against non-learnable operations and finally learnable operations surpass non-learnable ones. More comparison results of different cells are shown in Appendix.

5.3. Ablation Study

Bi-level V.S. Single-level In this paper, Single-DARTS optimizes network weights and architecture parameters with the same batch of data. However, there are also some different optimization methods, such as using two different batch of data from the train set. Here we conduct the ablation study on it and the results are shown in Table 4. The training phase lasts 50 epochs. w and α are optimized with the SGD optimizer. The learning rate is set to 0.005, which is the optimal value for these methods. We run each experiment 5 times and report the average values. We can see from the results that if we sample two different batch data from the same subset, the performance collapse is alleviated to some extent. Thus, using same batch of data is optimal, which is equivalent to single-level optimization in ours.

Different Learning Rates. We evaluate the performance of bi-level (DARTS) and single-level (Single-DARTS) optimization under different learning rates. Here, we use SGD as the optimizer and change the learning rate. Table 5 shows

Table 2: Comparison with SOTA architectures on ImageNet in DARTS space. †: the average results of searched architectures (not the average results of retraining searched the best architecture) and ∗: the best result. ⋆: scaling channels of the best such that its FLOPs bellow 600M. ⋄: adding extra training strategy on PDARTS setting.

Architecture	**FLOPs/M**	**Params/M**	**Top-1/%.**
NASNet-A [39]	564	5.3	74.0
AmoebaNet-C [30]	570	6.4	75.7
PDARTS [9]	557	4.9	75.6
PC-DARTS [34]	597	5.3	75.8
DARTS [27]	574	4.7	73.3
DARTS+⋄ [23]	591	5.1	76.3
CDARTS† [36]	732	6.1	76.3
CDARTS* [36]	704	6.3	76.6
CDARTS⋆ [36]	571	5.4	75.9
DropNAS⋄ [19]	597	5.4	76.6
Single-DARTS*	714	6.60	**77.0**
Single-DARTS†	707	6.55	76.7
Single-DARTS⋆	599	5.3	76.0

Table 3: Comparisons on CIFAR10 in DARTS space.

Architecture	**Params** (M)	**Top-1.** (%)
AmoebaNet-B [30]	2.8	97.45
PDARTS [9]	3.4	97.50
PC-DARTS [34]	3.6	$97.43_{\pm 0.07}$
DARTS [27]	3.3	$97.24_{\pm 0.09}$
DARTS+ ⋄ [23]	4.3	$97.63_{\pm 0.13}$
CDARTS [36]	3.8	$97.52_{\pm 0.04}$
DropNAS [19]	4.1	$97.42_{\pm 0.14}$
Single-DARTS	3.3	97.54

Table 4: Comparison of different optimization strategies with Softmax, whether to optimize on same subset or batch.

Subset	Batch	CIFAR10	CIFAR100	ImgNet_{16-120}
		$61.88_{\pm 10.72}$	$25.13_{\pm 13.47}$	$17.98_{\pm 2.35}$
✓		$84.34_{\pm 0.02}$	$54.92_{\pm 0.06}$	$25.97_{\pm 0.49}$
✓	✓	$\mathbf{94.27}_{\pm 0.13}$	$\mathbf{73.01}_{\pm 0.71}$	$\mathbf{46.10}_{\pm 0.34}$

that when the learning rate changes, the performance of DARTS varies dramatically, where the gap can be more than 50%. On the contrary, Single-DARTS shows much stronger stability and always achieves accuracies well above 93%.

Table 5: Accuracy under different learning rates (lr), activation functions (act), and optimizers (opt). * init α as 0.

Method	lr/act/opt	CIFAR10	CIFAR100
DARTS	0.001	$91.54_{\pm 1.26}$	$68.21_{\pm 1.15}$
	0.005	$61.88_{\pm 10.72}$	$25.13_{\pm 13.47}$
	0.025	$39.77_{\pm 0}$	$54.30_{\pm 0}$
Single-DARTS	0.001	$\mathbf{94.36}_{\pm 0}$	$\mathbf{73.51}_{\pm 0}$
	0.005	$\mathbf{94.36}_{\pm 0}$	$\mathbf{73.51}_{\pm 0}$
	0.025	$\mathbf{93.10}_{\pm 0}$	$\mathbf{69.24}_{\pm 0}$
DARTS	Softmax	$61.88_{\pm 10.72}$	$25.13_{\pm 13.47}$
	Sigmoid	$80.57_{\pm 0}$	$47.93_{\pm 0}$
Single-DARTS	Softmax	$\mathbf{94.27}_{\pm 0.13}$	$\mathbf{73.01}_{\pm 0.71}$
	Sigmoid	$\mathbf{94.36}_{\pm 0}$	$\mathbf{73.51}_{\pm 0}$
	Sigmoid*	$\mathbf{93.76}_{\pm 0}$	$\mathbf{70.71}_{\pm 0}$
DARTS	SGD	$61.88_{\pm 10.72}$	$25.13_{\pm 13.47}$
	Adam	$80.57_{\pm 0}$	$47.93_{\pm 0}$
Single-DARTS	SGD	$\mathbf{94.36}_{\pm 0}$	$\mathbf{73.51}_{\pm 0}$
	Adam	$\mathbf{94.36}_{\pm 0}$	$\mathbf{73.51}_{\pm 0}$

Different Activation Functions In Single-DARTS, we adopt a non-competitive activation function, Sigmoid, instead of the original Softmax function. Here we show the performance of DARTS and Single-DARTS under different activation functions in Table 5. We can see that using non-competitive activation functions is effective, which is consistent with the empirical results in previous works and the theoretical analysis in our paper. Moreover, Single-DARTS performs much more steadily than DARTS under different activation functions, no matter competitive or not.

Different Optimizers We evaluate the performance of DARTS and Single-DARTS under different optimizers. For SGD optimizer, we set momentum as 0.9, learning rate as 0.005, and weight decay as 3×10^{-4}. For Adam optimizer, we use a fixed learning rate of 3×10^{-4} and a momentum of (0.5, 0.999). From Table 5, using Adam as the optimizer can admittedly push the performance of DARTS to a higher level. On the contrary, Single-DARTS performs well no matter what optimizer it uses.

6. Conclusion

In this paper, we unveil that the performance collapse in DARTS is irreversible. Then we analyze this phenomenon from the perspective of optimization. Through gradient analysis, we state out that the bi-level framework plays a crucial role in performance collapse. On the basis of our theoretical insights, we propose a simple yet effective method, which uses single-level optimization. Ours outperforms previous methods both in accuracy and stability.

References

[1] Youhei Akimoto, Shinichi Shirakawa, Nozomu Yoshinari, Kento Uchida, Shota Saito, and Kouhei Nishida. Adaptive stochastic natural gradient method for one-shot neural architecture search. *arXiv preprint arXiv:1905.08537*, 2019. 2

[2] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. *arXiv preprint arXiv:1611.02167*, 2016. 1, 2

[3] Irwan Bello, Barret Zoph, Vijay Vasudevan, and Quoc V Le. Neural optimizer search with reinforcement learning. *arXiv preprint arXiv:1709.07417*, 2017. 2

[4] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In *International Conference on Machine Learning*, pages 549–558, 2018. 2

[5] Kaifeng Bi, Lingxi Xie, Xin Chen, Longhui Wei, and Qi Tian. Gold-nas: Gradual, one-level, differentiable. *arXiv preprint arXiv:2007.03331*, 2020. 2

[6] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Smash: one-shot model architecture search through hypernetworks. *arXiv preprint arXiv:1708.05344*, 2017. 2

[7] Han Cai, Chuang Gan, and Song Han. Once for all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019. 2

[8] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018. 2

[9] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. *arXiv preprint arXiv:1904.12760*, 2019. 2, 8

[10] Yukang Chen, Tong Yang, Xiangyu Zhang, Gaofeng Meng, Chunhong Pan, and Jian Sun. Detnas: Neural architecture search on object detection. *arXiv preprint arXiv:1903.10979*, 2019. 2

[11] Xiangxiang Chu, Xiaoxing Wang, Bo Zhang, Shun Lu, Xiaolin Wei, and Junchi Yan. Darts-: robustly stepping out of performance collapse without indicators. *arXiv preprint arXiv:2009.01027*, 2020. 7

[12] Xiangxiang Chu, Tianbao Zhou, Bo Zhang, and Jixiang Li. Fair darts: Eliminating unfair advantages in differentiable architecture search. *arXiv preprint arXiv:1911.12126*, 2019. 1, 2

[13] Xuanyi Dong and Yi Yang. One-shot neural architecture search via self-evaluated template network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3681–3690, 2019. 7

[14] Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1761–1770, 2019. 7

[15] Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. *arXiv preprint arXiv:2001.00326*, 2020. 1, 2, 11

[16] Yonggan Fu, Wuyang Chen, Haotao Wang, Haoran Li, Yingyan Lin, and Zhangyang Wang. Autogan-distiller: Searching to compress generative adversarial networks. *arXiv preprint arXiv:2006.08198*, 2020. 2

[17] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7036–7045, 2019. 2

[18] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. *arXiv preprint arXiv:1904.00420*, 2019. 2

[19] Weijun Hong, Guilin Li, Weinan Zhang, Ruiming Tang, Yunhe Wang, Zhenguo Li, and Yong Yu. Dropnas: Grouped operation dropout for differentiable architecture search. In *International Joint Conference on Artificial Intelligence*, 2020. 2, 8

[20] Andrew Hundt, Varun Jain, and Gregory D Hager. sharpdarts: Faster and more accurate differentiable architecture search. *arXiv preprint arXiv:1903.09900*, 2019. 2, 6

[21] Guilin Li, Xing Zhang, Zitong Wang, Zhenguo Li, and Tong Zhang. Stacnas: Towards stable and consistent optimization for differentiable neural architecture search. *arXiv preprint arXiv:1909.11926*, 2019. 1, 2

[22] Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. In *Uncertainty in Artificial Intelligence*, pages 367–377. PMLR, 2020. 7

[23] Hanwen Liang, Shifeng Zhang, Jiacheng Sun, Xingqiu He, Weiran Huang, Kechen Zhuang, and Zhenguo Li. Darts+: Improved differentiable architecture search with early stopping. *arXiv preprint arXiv:1909.06035*, 2019. 2, 8

[24] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L Yuille, and Li Fei-Fei. Autodeeplab: Hierarchical neural architecture search for semantic image segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 82–92, 2019. 2

[25] Chenxi Liu, Piotr Dollár, Kaiming He, Ross Girshick, Alan Yuille, and Saining Xie. Are labels necessary for neural architecture search? *arXiv preprint arXiv:2003.12056*, 2020. 6

[26] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 19–34, 2018. 1, 2

[27] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018. 1, 2, 7, 8

[28] Marcelo Gennari do Nascimento, Theo W Costain, and Victor Adrian Prisacariu. Finding non-uniform quantization schemes usingmulti-task gaussian processes. *arXiv preprint arXiv:2007.07743*, 2020. 2

[29] 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. 2, 7

[30] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture

search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789, 2019. 2, 8

[31] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2902–2911. JMLR. org, 2017. 2

[32] Richard Shin, Charles Packer, and Dawn Song. Differentiable neural network architecture search. 2018. 2

[33] Dimitrios Stamoulis, Ruizhou Ding, Di Wang, Dimitrios Lymberopoulos, Bodhi Priyantha, Jie Liu, and Diana Marculescu. Single-path nas: Device-aware efficient convnet design. *arXiv preprint arXiv:1905.04159*, 2019. 2

[34] 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. *arXiv preprint arXiv:1907.05737*, 2019. 1, 2, 6, 8

[35] Zhimin Xu, Si Zuo, Edmund Y Lam, Byoungho Lee, and Ni Chen. Autosegnet: An automated neural network for image segmentation. *IEEE Access*, 8:92452–92461, 2020. 2

[36] Hongyuan Yu and Houwen Peng. Cyclic differentiable architecture search. *arXiv preprint arXiv:2006.10724*, 2020. 6, 7, 8

[37] Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. Understanding and robustifying differentiable architecture search. *arXiv preprint arXiv:1909.09656*, 2019. 1

[38] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016. 2

[39] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018. 2, 8