IEEE International Conference on Computer Vision and Pattern Recognition (CVPR 2021)

DOTS: Decoupling Operation and Topology in Differentiable Architecture Search

Yu-Chao Gu[1*] Li-Juan Wang[1*] Yun Liu[1] Yi Yang[2] Yu-Huan Wu[1]
Shao-Ping Lu[1] Ming-Ming Cheng[1†]
[1]TKLNDST, CS, Nankai University [2]Zhejiang University

Abstract

*Differentiable Architecture Search (DARTS) has attracted extensive attention due to its efficiency in searching for cell structures. DARTS mainly focuses on the operation search and derives the cell topology from the operation weights. However, the operation weights can not indicate the importance of cell topology and result in poor topology rating correctness. To tackle this, we propose to **D**ecouple the **O**peration and **T**opology **S**earch (DOTS), which decouples the topology representation from operation weights and makes an explicit topology search. DOTS is achieved by introducing a topology search space that contains combinations of candidate edges. The proposed search space directly reflects the search objective and can be easily extended to support a flexible number of edges in the searched cell. Existing gradient-based NAS methods can be incorporated into DOTS for further improvement by the topology search. Considering that some operations (e.g., Skip-Connection) can affect the topology, we propose a group operation search scheme to preserve topology-related operations for a better topology search. The experiments on CIFAR10/100 and ImageNet demonstrate that DOTS is an effective solution for differentiable NAS. The code is released at* https://github.com/guyuchao/DOTS.

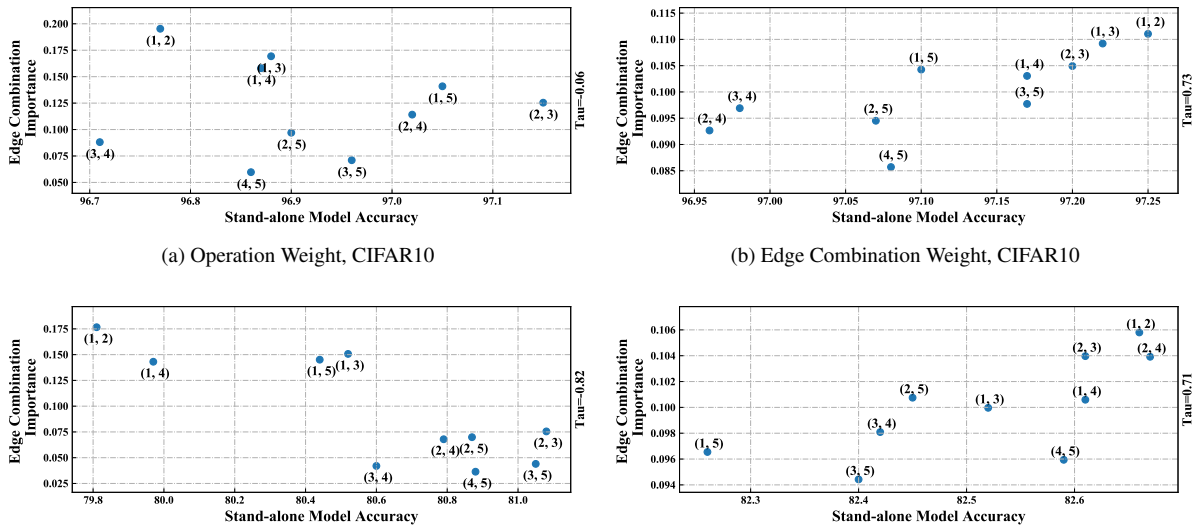
1. Introduction

Neural Architecture Search (NAS) has attracted extensive attention for its potential to find the optimal architecture in a large search space automatically. Previous reinforcement learning and evolutionary learning based approaches [34, 41, 58] require a full training process to validate the architecture performance, consuming hundreds of GPU-days to search. To reduce the search cost, one-shot methods [8, 10, 18, 40] adopt the weight sharing strategy, which trains the *supernet* once and derives child architecture performance from the supernet directly. Recent methods [4, 6, 50, 51] based on differentiable architecture search (DARTS) [35] also adopt the weight sharing strategy and further reduce the search cost by unifying the supernet training and child architecture searching.

In DARTS, the operation selection is parameterized with learnable operation weights, which is updated with the supernet training. After training, the operation weights are used to rank the importance of operations and topology. The edge importance in DARTS is represented as the largest operation weight on this edge. DARTS retains the two most important edges for each intermediate node to derive the topology of the searched cell. A question is raised: whether the edge importance indicated by the operation weights accurately ranks the stand-alone model's performance. As illustrated in Fig. 1, we find no obvious rank correlation, which implies that DARTS has no superiority over choosing edges randomly (see more details in Sec. 3.2). Furthermore, DARTS' handcraft policy of edge numbers restricts their potential to find more flexible cell structures.

This paper addresses the above problems via **D**ecoupling **O**peration and **T**opology **S**earch (DOTS). The meaning of decoupling is two-fold. On the one hand, we decouple the topology representation from the operation weights. In detail, we introduce a topology search space containing the pairwise combinations of edges. The topology search space is continuously relaxed, and the relaxed topology weights model the combinatorial distribution of candidate edges. The proposed topology search space directly reflects the search objective and can be easily extended to support a flexible number of edges. On the other hand, we decouple the operation and topology search processes. The overall search process is divided into the operation search stage and the topology search stage, in which we update operation weights and edge combination weights, respectively. With decoupling the two searching processes, existing gradient-based NAS methods can be directly incorporated into the DOTS' operation search and get further improvement by the topology search. Furthermore, the topology search is performed in a shrunk supernet, making it more efficient

*Both authors contributed equally to this work.
†M.M. Cheng (cmm@nankai.edu.cn) is the corresponding author.



(a) Operation Weight, CIFAR10

(b) Edge Combination Weight, CIFAR10

(c) Operation Weight, CIFAR100

(d) Edge Combination Weight, CIFAR100

Figure 1: Rank correlation analysis between different edge importance representations and the stand-alone model performance. The edge combination importance is indicated by operation weights (DARTS) and edge combination weights (DOTS). We calculate the *Kendall Tau* metric [25] to measure the rank correlation.

and accurate. Considering that some operations (*e.g.*, *Skip-Connection*) can affect the topology, we adopt a group strategy in the operation search to preserve these topology-related operations for a better topology search.

We summarize our contributions as follows:

- We propose to decouple the operation and topology search, which decouples both the topology representation and search processes. Such decoupling leads to the correct rating of stand-alone models with different topologies.
- The proposed topology search space can be extended to support a flexible number of edges in the searched cell, fulfilling its potential to search for more complex structures.
- Existing gradient-based methods can be incorporated into DOTS and get further improvement by the topology search. DOTS only costs 0.26 and 1.3 GPU-days to search from scratch and achieves 97.51% and 76.0% accuracy on CIFAR10 and ImageNet. Better performance can be achieved if the constraint of edge numbers is removed.

2. Related Work

Different from the previous manually designing task-specific neural networks [14, 17, 36, 46, 47], Neural Architecture Search (NAS) has attracted extensive attention for its potential to design efficient networks automatically [15, 16, 30, 32]. Early methods based on reinforcement learning [57, 58] and evolutionary algorithms [41, 48] train thousands of candidate architectures from scratch and use their validation accuracy to learn a meta-controller, which requires prohibitive search cost. Recent one-shot NAS methods [2, 3, 8, 18] and gradient-based methods [9, 19, 35, 50] adopt the weight sharing strategy [40], which only trains the *supernet* once and thus reduces the search cost. Recent gradient-based methods try to overcome the instability [5,7,9,19,31,50,53] and reduce the search cost [6,11,51]. Previous gradient-based methods mainly target improving the operation search. While our work improves gradient-based methods with the topology search, which is complementary to previous researches.

Recent researches [12, 13, 42, 49] reveal the importance of connection topology in neural networks. The randomly-wired network [49] finds that networks generated by random graph algorithms can obtain competitive results. Shu *et al.* [42] find that the cell topology has more impact on network convergence than the operation in cell-based NAS. DenseNAS [12] proposes a densely-connected search space that focuses on the macro structure's topology. Our work sheds light on the micro cell's topology and constructs a topology search space to support a flexible number of edges.

Recent weight sharing methods [22, 26, 27, 29, 52, 55, 56] try to improve the architecture rank correctness. Yu *et al.* [52] point out that recent NAS has similar performance to random search because of the constrained search space and the widely-used weight sharing strategy. PCNAS [29] identifies and fixes the posterior fading problem in weight-sharing methods. Block-wisely NAS [26] improves the architecture rank correctness by modularizing the large space

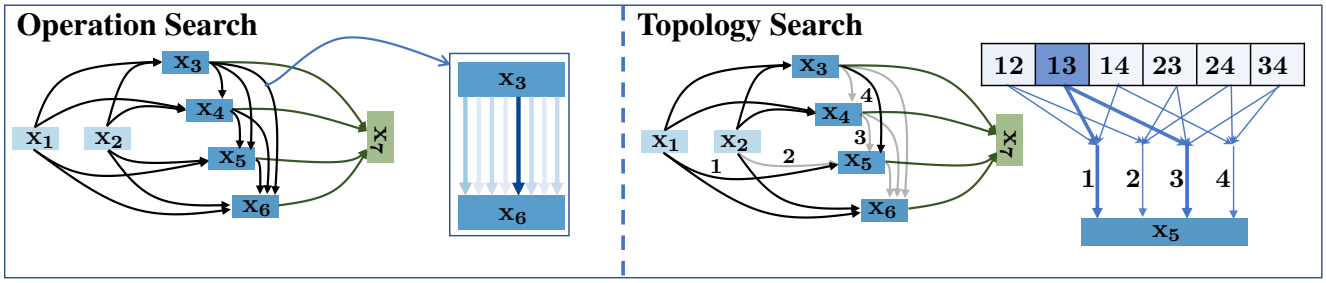


Figure 2: Overall pipeline of the proposed DOTS. The DOTS framework consists of the operation search and the topology search. In the operation search phase, we search for the best performing operations on each edge. And in the topology search phase, we search for the best combination for candidate edges.

into blocks. Although some recent works notice the rating problem in one-shot methods, there is less concern about the rating problem in gradient-based methods.

3. Review of DARTS

3.1. Preliminary of DARTS

We start by reviewing the baseline algorithm DARTS [35]. DARTS aims at searching for the cell, a repeating building block of the neural network. A cell is represented as a directed cyclic graph with N nodes $\{x_i\}_{i=1}^{N}$, including two input, one output, and several intermediate nodes. Each node denotes a feature map transformed by graph edges. The j-th intermediate node x_j connects to all its predecessors x_i through the edge (i, j). Each edge (i, j) contains candidate operations weighted by the operation weight $\alpha^{(i,j)}$, which can be defined as

$$\bar{o}^{(i,j)}(x) = \sum_{o \in O} \alpha_o^{(i,j)} o^{(i,j)}(x_i), \tag{1}$$

where $o(x) \in \mathcal{O}$ and $\mathcal{O}$ is the operation search space containing eight operations, including *Zero*, *Skip-Connection*, *Avg-Pooling*, *Max-Pooling*, *Sep3x3Conv*, *Sep5x5Conv*, *Dil3x3Conv*, and *Dil5x5Conv*. The weight for each operation is normalized with softmax:

$$\alpha_o^{(i,j)} = \frac{\exp(\alpha'^{(i,j)}_o)}{\sum_{o' \in \mathcal{O}} \exp(\alpha'^{(i,j)}_{o'})}, \tag{2}$$

where α' is the unnormalized operation weight. The operation weight is updated with the supernet training, gradually focusing on the optimal architecture. Once defined the mixed operation $\bar{o}^{(i,j)}$ for edge (i, j), the intermediate node x_j is computed from all its predecessors x_i:

$$x_j = \sum_{i<j} \bar{o}^{(i,j)}(x_i). \tag{3}$$

Let $\mathcal{L}_{cls}^{train}$ and $\mathcal{L}_{cls}^{val}$ be the Cross-Entropy loss on the training and validation sets, respectively. Then, we can formulate the bi-level optimization with the operation weight α and the network weight w as

$$\begin{aligned} &\min_{\alpha} \ \mathcal{L}_{cls}^{val}(w^*(\alpha), \alpha), \\ &\text{s.t.} \ \ w^*(\alpha) = \arg\min_{w}(\mathcal{L}_{cls}^{train}(w, \alpha)). \end{aligned} \tag{4}$$

After searching, the final architecture is derived from the operation weight α by two hard pruning:

1. Retain the operation with the largest weight and prune other operations for each edge, *i.e.*, $o^{(i,j)} = \arg\max_{o \in \mathcal{O}, o \neq Zero} \alpha_o^{(i,j)}$.
2. Retain two incoming edges with the largest edge importance for each intermediate node and prune other edges. The edge importance is defined as the largest operation weight on each edge (i, j), *i.e.*, $\max_{o \in \mathcal{O}, o \neq Zero} \alpha_o^{(i,j)}$.

3.2. Coupling Problem in DARTS

Previous works [5–7, 11, 20, 28, 31, 38, 50, 51] based on DARTS indicate the edge importance by the largest operation weight (excluding the *Zero* operation) on this edge. We conduct a rank correlation analysis to find whether edge importance indicated by operation weights can accurately rank the stand-alone model (measured by *Kendall Tau* metric [25]). We follow DARTS' handcraft policy that sums the importance of two edges to get its edge combination importance. However, for DOTS, the edge combination weights can directly represent the edge combination importance. There are five edges in our experiment, resulting in ten different edge combinations. The stand-alone model is trained with the same setting as in Sec. 5.1, except that we reduce the number of training epochs to 300.

Fig. 1a and Fig. 1c show that the stand-alone model accuracy has no clear ranking correlation with edge importance indicated by the operation weights. It implies that DARTS' searched cell is sub-optimal because it cannot converge to the best cell topology, which is consistent with the finding that DARTS cannot surpass random search in some cases. Intuitively, the larger operation weight can only indicate an

operation's suitability for a specific edge, but does not mean the edge should be retained in the searched topology. As shown in Fig. 1b and Fig. 1d, the proposed edge combination weights achieve $Tau = 0.73$ and $Tau = 0.71$ on CIFAR10 and CIFAR100, demonstrating its effectiveness for edge selection.

4. Methodology

The above analysis points out the limitation of coupling the operation and topology search. In this section, we try to tackle this problem via decoupling operation and topology search. As shown in Fig. 2, the overall search is divided into the operation search and the topology search. In the operation search phase, we search for the best operation on each edge. In the topology search phase, we search for the best combination of candidate edges. In Sec. 4.1, we introduce how to construct the topology search space and support a flexible number of edges. In Sec. 4.2, we describe how to incorporate existing gradient-based NAS methods into DOTS' operation search and propose a group operation search scheme to retain topology-related operations for a better topology search.

4.1. Topology Search

4.1.1 Handcraft Policy for the Number of Edges

In Sec. 3.2, we have discussed the limitation of coupling the operation and topology search. Hence, we need to decouple the edge importance from operation weights. To achieve this, we define a topology search space apart from the operation search space.

Formally, the j-th intermediate node x_j connects to all its predecessors x_i through the edge (i, j). Following DARTS' handcraft policy to restrict two edges for intermediate nodes, we make the topology search space $\mathcal{E}_{x_j}$ for x_j as all pairwise combinations of its incoming edges:

$$\mathcal{E}_{x_j} = \{\langle (i_1, j), (i_2, j)\rangle \,|0 < i_1 < i_2 < j\}. \tag{5}$$

Suppose that there are total n incoming edges for node x_j, so the search space $\mathcal{E}_{x_j}$ contains $C_n^2 = \frac{n!}{2!(n-2)!}$ candidate combinations. For each node x_j, we relax its topology search space to continuous, which can be defined as

$$\beta_{x_j}^c = \frac{\exp(\beta'^{x_j}_c / T_\beta)}{\sum_{c' \in \mathcal{E}_{x_j}} \exp(\beta'^{x_j}_{c'} / T_\beta)}, \tag{6}$$

where $\beta_{x_j}^c$ denotes the normalized probability of choosing edge combination c. Although the topology search space is defined on edge combinations, we do not need to obtain each edge combination feature in practice. To reduce the memory cost, we aggregate weight for edge e_j^i from those combinations containing this edge, which can be formulated as

$$\gamma^{(i,j)} = \sum_{c \in \mathcal{E}_{x_j}, (i,j) \in c} \frac{1}{N(c)} \beta_{x_j}^c, \tag{7}$$

where $\gamma^{(i,j)}$ is the weight of each edges and $N(c)$ is the edges number in edge combination c. We sum all the incoming edges of x_j weighted by the edge importance weight γ to get its features:

$$x_j = \sum_{i<j} \gamma^{(i,j)} \cdot \bar{o}^{(i,j)}(x_i), \tag{8}$$

where $\bar{o}^{(i,j)}$ means the mixed operation on edge (i, j). Since we decouple the operation and topology search processes, the $\bar{o}^{(i,j)}$ mixes the candidate operations retained by the operation search, which will be discussed in Sec. 4.2.

As discussed in ASAP [38] and SNAS [50], the optimization gap between the supernet and the derived child network causes a performance drop. Both works exploit architecture annealing to bridge the optimization gap during searching. We generalize the annealing idea to the topology search. In Equ. (6), T_β is the annealing temperature. We adopt an exponential schedule for annealing:

$$T(t) = T_0 \theta^t, \tag{9}$$

where it starts from an initial temperature T_0 and decays with the training step t increasing.

DARTS uses bi-level optimization to avoid overfitting [35]. However, [18, 28] shows that one-level optimization is stable and accurate. In our topology search stage, the operation on each edge is largely reduced, eliminating the risk of overfitting. Therefore, we use one-level optimization for updating the network weight w and the topology weight β, which can be formulated as

$$\begin{aligned} w_t &= w_{t-1} - \eta_t \partial_w L_{train}(w_{t-1}, \beta_{t-1}), \\ \beta_t &= \beta_{t-1} - \delta_t \partial_w L_{train}(w_{t-1}, \beta_{t-1}), \end{aligned} \tag{10}$$

where η_t and δ_t are the learning rates of the network weight and topology weight, respectively.

4.1.2 Flexible Number of Edges

In Sec. 4.1.1, we construct a topology search space following the DARTS handcraft policy to restrict each intermediate node in the searched cell connects two edges. Such a handcraft policy cannot learn the number of edges automatically. Hence, we extend the topology search space to support arbitrary numbers of edges. Specifically, we adopt binary code to describe such search space. For the intermediate node x_j with n incoming edges, the binary code for the m-th edge combination can be represented as

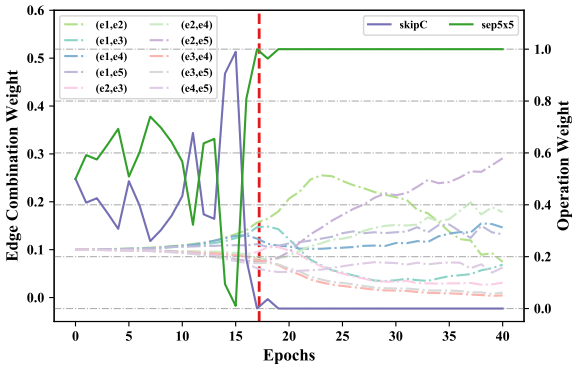


Figure 3: Topology search with more than one operation on each edge. We adopt a lower initial temperature to anneal operation weight than topology weight, *i.e.*, $T_{\alpha_{o_n}} = \frac{1}{1000} T_\beta$. Hence, the operation is fixed on each edge in the first few epochs (denoted by the red dash line).

$$c_m = \{e_1, e_2, \dots, e_n\}, \tag{11}$$

where $e_i \in \{0, 1\}$ denotes the edge i exists or not in an edge combination. The topology search space for the node x_j can be defined as

$$\mathcal{E}_{x_j} = \{c^1, c^2, \dots, c^M\}, \tag{12}$$

where M is the number of valid edge combinations. We can compute M by

$$M = \sum_{k=1}^{n} C_n^k = 2^n - 1. \tag{13}$$

We exclude the extreme case where all edges are not in the edge combination. Overall, we enable searching for a flexible number of edges. This is easy to implement as we only replace the search space of Equ. (5) with Equ. (12) and keep the architecture relaxation and optimization the same.

4.2. Operation Search

In this section, we introduce the operation search of DOTS. The operation search aims to retain the optimal operation candidate on each edge. We introduce how to incorporate the existing gradient-based NAS methods into DOTS' operation search in Sec. 4.2.1. Existing methods only retain the best operation on each edge, discarding that some operations (*e.g.*, *Skip-Connection*) may affect the topology. To prevent topology-related operations from being pruned in the operation search, we propose a group operation search scheme in Sec. 4.2.2.

4.2.1 Incorporating Gradient-Based NAS Methods

The previous gradient-based methods based on DARTS can be easily incorporated into DOTS' operation search. DARTS introduces a set of weights $\alpha = \{\alpha^{(i,j)}\}$ for candidate operations on each edge (i, j). We follow their own searching strategies to get the trained operation weight α. Then, the operation with the maximum weight is retained on each edge (i, j):

$$o^{(i,j)} = \arg\max_{o \in \mathcal{O}} \alpha_o^{(i,j)}. \tag{14}$$

The final step is to replace the mixed operation $\bar{o}^{(i,j)}$ in Equ. (8) with $o^{(i,j)}$. Overall, existing gradient-based NAS methods can be easily plugged into DOTS' operation search and get further improvement by the topology search.

4.2.2 Operation Search with Group Strategy

Generally, the operations can be categorized into topology-related (*e.g.*, *Skip-Connection*) and topology-agnostic (*e.g.*, *Separable Convolution*). In Sec. 4.2.1, we incorporate the existing gradient-based methods into DOTS' operation search, where the best operation is retained on each edge. Such a strategy eliminates potential topology choices because some topology-related operations are dropped before the topology search. To this end, we resort to group strategy [20, 28] for the operation search. The group strategy can ensure to retain both topology-related and topology-agnostic operations for the topology search.

Formally, in the group operation search, the operation search space $\mathcal{O}$ is divided into several subspaces $\mathcal{O} = \{\mathcal{O}_1, \mathcal{O}_2, \dots \mathcal{O}_p\}$, where p is the number of groups. Each operation subspace is relaxed to continuous independently. After searching, the operation with the largest weight is chosen from each group to construct a new operation search space $\mathcal{O}_n$ on each edge (i, j), which can be formulated as

$$o_p^{(i,j)} = \arg\max_{o_p \in \mathcal{O}_p} \alpha_{o_p}^{(i,j)}, \tag{15}$$

$$\mathcal{O}_n = \{o_1^{(i,j)}, o_2^{(i,j)}, \dots, o_p^{(i,j)}\}. \tag{16}$$

We evaluate different group strategies and discuss them in the experiment. Since O_n contains more than one operation, we need to search for it in the topology search stage. Simultaneously updating the operation and topology is inaccurate because it increases the weight-sharing child models [8, 29]. To tackle this problem, we anneal the operation weight with a lower temperature $T_{\alpha_{o_n}}$ than T_β in the topology search. As illustrated in Fig. 3, the operations in two groups are fixed in the first few epochs. Once the operations are fixed, further topology search evolves similarly as that in Sec. 4.2.1.

5. Experiment

5.1. Evaluation on CIFAR10/100

Search Settings. We implement the DOTS based on Pytorch [39], Mindspore [1] and Jittor [23] frameworks. The

Architecture	Top-1 Acc. (%) CIFAR10	Params (M) CIFAR10	Top-1 Acc. (%) CIFAR100	Params (M) CIFAR100	Search Cost (GPU-days)	Search Method
DenseNet-BC [24]	96.54	25.6	82.82†	25.6	N/A	N/A
NASNet-A [58]	97.35	3.3	83.18	3.3	1800	RL
ENAS [40]	97.11	4.6	80.57†	4.6	0.5	RL
AmoebaNet-B [41]	97.45± 0.05	2.8	-	-	3150	EA
Hireachical Evolution [34]	96.25± 0.12	15.7	-	-	300	EA
PNAS [33]	96.59± 0.09	3.2	80.47†	3.2	225	SMBO
DARTS [35]	97.00	3.4	82.46†	3.4	0.4	GD
SNAS [50]	97.15	2.8	82.45	2.8	1.5	GD
GDAS [11]	97.07	2.5	81.62†	3.4	0.2	GD
P-DARTS [6]	97.50	3.4	82.51†	3.6	0.3	GD
FairDARTS [9]	97.46	2.8	82.39	2.8	0.4	GD
PC-DARTS [51]	97.43 ± 0.07	3.6	83.10	3.6	0.1	GD
DropNAS [20]	97.42 ± 0.14	4.1	83.13	4.0	0.6	GD
MergeNAS [45]	97.27 ± 0.02	2.9	82.42	2.9	0.2	GD
ASAP [38]	97.32 ± 0.11	2.5	82.69	2.5	0.2	GD
SDARTS-ADV [5]	97.39 ± 0.02	3.3	83.27	3.3	1.3	GD
DARTS- [7]	97.41 ± 0.08	3.5	82.84†	3.4	0.4	GD
DOTS (best)	97.63	3.5	83.72	4.1	0.26	GD
DOTS (avg)*	97.51±0.06	3.5	83.52±0.13	4.1	0.26	GD

Table 1: Comparison with state-of-the-art models on CIFAR10/100. CIFAR10 evaluation results are taken from their original paper. †: The CIFAR100 results are reported by Chu et al [7]. *: Our results are obtained by four individual runs of search and evaluation under different random seeds.

Architecture	TS	CIFAR10	CIFAR100
DARTS [35]	✗	97.02±0.12	80.74
	✓	97.40±0.09	83.07
DARTS (2nd) [35]	✗	97.01±0.15	81.37
	✓	97.12±0.11	83.28
GDAS [11]	✗	96.84±0.06	82.75
	✓	97.06±0.08	83.01
SNAS [50]	✗	97.05±0.10	81.92
	✓	97.26±0.12	83.25
PC-DARTS [51]	✗	97.28±0.08	81.74
	✓	97.45±0.06	82.36

Table 2: Improving existing gradient-based NAS methods by the topology search. TS means the topology search.

whole search process on CIFAR10/100 takes 70 epochs, *i.e.*, 30 epochs for the operation search and 40 for the topology search. The network is composed of 8 cells for the operation search and 20 cells for the topology search. The initial temperature is set to $T_0 = 10$ and decay to 0.02 in the topology search. The search process costs 6.3 hours (0.26 GPU-days) on one NVIDIA Tesla V100 GPU. More detailed search settings can be found in the supplementary.

Evaluation Settings. The evaluation network is composed of 20 cells (18 normal cells and 2 reduction cells) and the initial number of channels is 36. We train the network from scratch for 600 epochs with a batch size of 96. The network is optimized via the SGD optimizer with an initial learning rate of 0.025 (cosine annealing to 0), momentum of 0.9, weight decay of $3e$-4, and gradient clipping at 5. Cutout and drop-path with a rate of 0.2 are used for preventing overfitting.

Main Results. The evaluation results on CIFAR10/100 are shown in Tab. 1. DOTS only costs 0.26 GPU-days to achieve 97.51% and 83.72% accuracy on CIFAR10 and CIFAR100, respectively. Thanks to the decoupling of the operation and topology search, the number of candidate operations on edges is largely reduced, and both stages are fast to converge. DOTS improves DARTS by 0.51% on CIFAR10 with lower search costs. The previous effort in gradient-based methods can be incorporated into the DOTS framework and further improved by the topology search, which is shown in Tab. 2.

5.2. Evaluation on ImageNet

Search Settings. The whole search process takes 70 epochs, 30 epochs for the operation search and 40 epochs for the topology search. The network is composed of 14 cells for both search stages. The initial temperature T_0 is set to 10 and decay to 0.02 in the topology search. The whole search process costs 7.8 hours (1.3 GPU-Days) on four NVIDIA Quadro RTX 8000 GPUs. Other search settings we keep the same as PC-DARTS [51], which can be found in supplementary.

Evaluation Settings. The evaluation follows the mobile setting, where the input image size is set to 224×224, and the number of multiply-add operations is restricted to be fewer than 600M. The network consists of 14 cells (12 normal cells and 2 reduction cells) with an initial number of channels of 46. We train the network from scratch for 250 epochs with a batch size of 1024. The SGD optimizer with an initial learning rate of 0.5 (warm up in the first 5 epochs and cosine annealing to 0), momentum of 0.9, and weight decay of $3e$-5 is used. Additional enhancements follow P-DARTS [6] and PC-DARTS [51], including label smooth-

Architecture	Acc. (%)		Params	Multi-Add	Search Cost	Search Method
	top-1	top-5	(M)	(M)	(GPU-days)	
Inception-v1 [43]	69.8	89.9	6.6	1448	N/A	N/A
MobileNet [21]	70.6	89.5	4.2	569	N/A	N/A
ShuffleNet 2× (v1) [54]	73.6	89.8	5.4	524	N/A	N/A
ShuffleNet 2× (v2) [37]	74.9	92.4	7.4	591	N/A	N/A
NASNet-A [58]	74.0	91.6	5.3	564	1800	RL
AmoebaNet-C [41]	75.7	92.4	6.4	570	3150	EA
PNAS [33]	74.2	91.9	5.1	588	225	SMBO
MnasNet-92 [44]	74.8	92.0	4.4	388	1667	RL
DARTS (2nd order) (CIFAR10) [35]	73.3	91.3	4.7	574	4.0	GD
SNAS (CIFAR10) [50]	72.7	90.8	4.3	522	1.5	GD
P-DARTS (CIFAR10) [6]	75.6	92.6	4.9	557	0.3	GD
GDAS (CIFAR10) [11]	74.0	91.5	5.3	581	0.2	GD
FairDARTS (CIFAR10) [9]	75.1	92.5	4.8	541	0.4	GD
PC-DARTS (CIFAR10) [51]	74.9	92.2	5.3	586	0.1	GD
SDARTS-ADV (CIFAR10) [5]	74.8	92.2	5.4	594	1.3	GD
DropNAS (CIFAR10) [20]	75.5	92.6	5.2	572	0.6	GD
ASAP (CIFAR10) [38]	73.7	91.5	3.8	427	0.2	GD
DOTS (CIFAR10)	75.7	92.6	5.2	581	0.2	GD
ProxylessNAS (ImageNet) [4]	75.1	92.5	7.1	465	8.3	GD
FairDARTS (ImageNet) [9]	75.6	92.6	4.3	440	3	GD
PC-DARTS (ImageNet) [51]	75.8	92.7	5.3	597	3.8	GD
DOTS (ImageNet)	76.0	92.8	5.3	596	1.3	GD

Table 3: Comparison with state-of-the-art models on ImageNet. CIFAR10 and ImageNet mean the cell architecture is searched on CIFAR10 or ImageNet.

Topology Parameterized Strategy	Edge Numbers Constraint	CIFAR10 Test Acc. (%)	CIFAR100 Test Acc. (%)
PC-DARTS	2	97.38 ±0.09	82.98
DOTS	2	97.51 ±0.06	83.72
Edge-Level Sigmoid	arbitrary	97.26 ±0.14	81.02
DOTS	arbitrary	97.53 ±0.08	83.92

(a) Ablation study of different topology parameterized strategy.

Operation Search Strategy	#op	CIFAR10 Test Acc. (%)	CIFAR100 Test Acc. (%)	Search Cost (GPU-days)
DARTS-Top1	1	97.40 ±0.09	83.07	0.22
DARTS-Top2	2	97.42 ±0.11	82.96	0.26
Group-V1 [28]	4	97.48 ±0.11	83.55	0.35
Group-V2 [20]	2	97.51 ±0.06	83.72	0.26

(b) Ablation study of different operation search strategies. #op: The numbers of operation retained on each edge.

Table 4: Ablation studies of different strategies.

ing and an auxiliary loss tower.

Main Results. The evaluation results are summarized in Tab. 3. Most gradient-based methods search on a proxy dataset, *e.g.*, CIFAR10, and transfer the searched cell to ImageNet because their search cost on ImageNet is prohibitive. While DOTS can search on ImageNet proxylessly, and requires the least search costs (1.3 GPU-days). DOTS improves DARTS by 2.7% of top-1 accuracy on ImageNet.

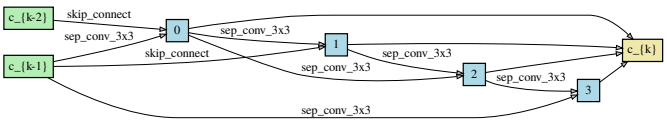
5.3. Ablation Study

Improving gradient-based NAS methods. We incorporate five gradient-based methods into DOTS and validate the effectiveness of the topology search. The baseline is to derive the topology of the searched cell by DARTS policy. Comparison results are summarized in Tab. 2. We can observe that adding topology search results in a consistent improvement over the baseline, verifying the effectiveness of the proposed topology search. An illustrative example is shown in Fig. 4. We use DARTS for the operation search and the operation search results are shown in Fig. 4a. Directly deriving its topology based on the operation weights, *i.e.*, DARTS' policy, results in the searched cell in Fig. 4c. We can find it is dominated by skip-connection and only achieves 80.74% accuracy on CIFAR100. Fig. 4b shows the topology search results based on the same searched operations, which obtains a 2.33% improvement (83.07% *vs.* 80.74%). We can find the topology search helps remedy the unstable results of the operation search by further choosing edge connections.

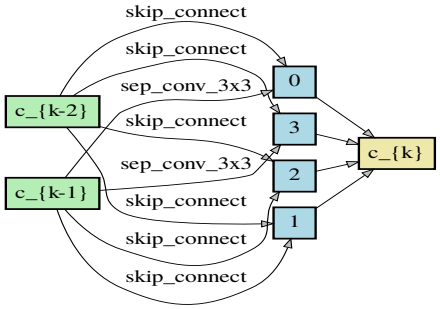
Influence of topology parameterized strategy. We validate the effectiveness of the proposed edge combination weights and compare it with some other topology parameterized strategies. First, we compare with PC-DARTS that directly adds learnable weight on edges. Since it is not easy for the PC-DARTS strategy to support flexible edge numbers in the searched cell, we restrict both methods to retain two edges per node after the topology search. From the results in Tab. 4a, the proposed DOTS improves the edge weight of PC-DARTS by 0.13% on CIFAR10. This mainly because the original purpose of the edge weight in PC-DARTS is to stabilize the training, not for the topology search. In comparison, the proposed edge combination

EdgeID	1	2	3	4	5	6	7
OP	SkipC	Sep3x3	SkipC	SkipC	Sep3x3	SkipC	SkipC
EdgeID	8	9	10	11	12	13	14
OP	Sep3x3	Sep3x3	SkipC	Sep3x3	Sep3x3	Sep3x3	Sep3x3

(a) Operation searched results



(b) DOTS topology search

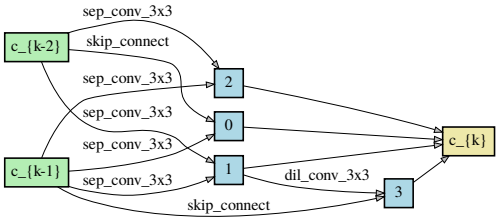


(c) DARTS' policy

Figure 4: Results of different topology derivations. (a) The operations searched by DARTS on CIFAR10. (b) Deriving topology based on proposed topology search. (c) Deriving topology based on DARTS' policy.

EdgeID	1	2	3	4	5	6	7
OP Group 1 OP Group 2	SkipC Dil3x3	SkipC Sep3x3	SkipC Sep3x3	SkipC Sep3x3	SkipC Sep3x3	SkipC Sep3x3	SkipC Sep3x3
EdgeID	8	9	10	11	12	13	14
OP Group 1 OP Group 2	SkipC Dil3x3	SkipC Dil3x3	SkipC Dil5x5	SkipC Sep3x3	SkipC Sep3x3	SkipC Dil3x3	Zero Sep3x3

(a) Operation searched results



(b) Topology searched results

Figure 5: Results of the operation and topology search based on Group-V2 strategy.

weights can directly reflect the objective of edge selection.

Then we validate the effectiveness of DOTS in arbitrary edge numbers setting. By removing the edge number constraint, the performance of DOTS is promoted from 83.72% to 83.92% on CIFAR100. In the arbitrary edge numbers setting, a straightforward method is to replace the softmax activation with sigmoid activation on the edge weights and binarize the choice with a threshold. We name this strategy edge-level sigmoid. From Tab. 4a, the DOTS has a clear advantage over the edge-level sigmoid strategy.

Influence of the operation search strategy. We investigate the different operation search strategies and discuss their influence to the topology search. The baseline is DARTS-Top1, which retains one strongest operation on each edge for the topology search. The operation search with Group-V2 strategy (details in supplementary) improves DARTS-Top1 by 0.11% and 0.65% on CIFAR10 and CIFAR100, respectively. The reason is DARTS-Top1 overlooks a case that some operations are topology-related. Pruning these operations in the operation search stage will eliminate the potential topology choices, resulting in sub-optimal solutions.

An illustrative example of the group strategy is shown in Fig. 5. From Fig. 5a, the best operation in the topology-related and topology-agnostic group is retained for the topology search. Fig. 5b shows the topology search results based on the operation search results in Fig. 5a. Direct retaining more operations, *i.e.*, DARTS-Top2 has no obvious improvement, because it does not guarantee topology-related operations are preserved. Adding more groups in topology-agnostic operations *i.e.*, Group-V1 strategy has no improvement but increases the search cost. From this experiment, we can find preserve two operations (one topology-related and one topology-agnostic) is sufficient for the topology search.

6. Conclusion

In this paper, we study the topology derivation of existing gradient-based methods based on DARTS. The edge importance of DARTS is based on the operation weights, which can not correctly rank the stand-alone models with different topologies. Thus we propose to decouple the topology representation from the operation weights and make an explicit topology search. The proposed topology representation, *i.e.*, edge combination weights, leads to the correct topology rating and supports flexible edge numbers. Apart from decoupling the operation and topology representation, we propose to decouple their searching processes to make an efficient and accurate topology search. Experiments on CIFAR and ImageNet demonstrate DOTS is an efficient and effective solution to differentiable NAS.

Acknowledgement This research was supported by the Major Project for New Generation of AI under Grant No. 2018AAA0100400, S&T innovation project from Chinese Ministry of Education, and NSFC (61922046). We thank MindSpore for the partial support of this work.

References

[1] Mindspore. https://www.mindspore.cn/, 2020. 5

[2] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc V. Le. Understanding and simplifying one-shot architecture search. In *ICML*, pages 549–558, 2018. 2

[3] Andrew Brock, Theo Lim, J.M. Ritchie, and Nick Weston. SMASH: One-shot model architecture search through hypernetworks. In *ICLR*, 2018. 2

[4] Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Dsearchirect neural architecture search on target task and hardware. In *ICLR*, 2019. 1, 7

[5] Xiangning Chen and Cho-Jui Hsieh. Stabilizing differentiable architecture search via perturbation-based regularization. In *ICML*, 2020. 2, 3, 6, 7

[6] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *ICCV*, pages 1294–1303, 2019. 1, 2, 3, 6, 7

[7] 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. 2, 3, 6

[8] Xiangxiang Chu, Bo Zhang, Ruijun Xu, and Jixiang Li. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. *arXiv preprint arXiv:1907.01845*, 2019. 1, 2, 5

[9] Xiangxiang Chu, Tianbao Zhou, Bo Zhang, and Jixiang Li. Fair darts: Eliminating unfair advantages in differentiable architecture search. *arXiv preprint arXiv:1911.12126*, 2019. 2, 6, 7

[10] Xuanyi Dong and Yi Yang. One-shot neural architecture search via self-evaluated template network. In *ICCV*, pages 3681–3690, 2019. 1

[11] Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. In *CVPR*, pages 1761–1770, 2019. 2, 3, 6, 7

[12] Jiemin Fang, Yuzhu Sun, Qian Zhang, Yuan Li, Wenyu Liu, and Xinggang Wang. Densely connected search space for more flexible neural architecture search. In *CVPR*, pages 10628–10637, 2020. 2

[13] Shanghua Gao, Ming-Ming Cheng, Kai Zhao, Xin-Yu Zhang, Ming-Hsuan Yang, and Philip HS Torr. Res2net: A new multi-scale backbone architecture. *IEEE TPAMI*, 2019. 2

[14] Shang-Hua Gao, Qi Han, Duo Li, Pai Peng, Ming-Ming Cheng, and Pai Peng. Representative batch normalization with feature calibration. In *CVPR*, 2021. 2

[15] Shang-Hua Gao, Qi Han, Zhong-Yu Li, Pai Peng, Liang Wang, and Ming-Ming Cheng. Global2local: Efficient structure search for video action segmentation. In *CVPR*, 2021. 2

[16] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *CVPR*, pages 7036–7045, 2019. 2

[17] Yuchao Gu, Lijuan Wang, Ziqin Wang, Yun Liu, Ming-Ming Cheng, and Shao-Ping Lu. Pyramid constrained self-attention network for fast video salient object detection. In *AAAI*, volume 34, pages 10869–10876, 2020. 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. 1, 2, 4

[19] Chaoyang He, Haishan Ye, Li Shen, and Tong Zhang. Milenas: Efficient neural architecture search via mixed-level reformulation. In *CVPR*, pages 11993–12002, 2020. 2

[20] Weijun Hong, Guilin Li, Weinan Zhang, Ruiming Tang, Yunhe Wang, Zhenguo Li, and Yong Yu. DropNAS: Grouped operation dropout for differentiable architecture search. In *IJCAI*, pages 2326–2332, 2020. 3, 5, 6, 7

[21] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 7

[22] Shoukang Hu, Sirui Xie, Hehui Zheng, Chunxiao Liu, Jianping Shi, Xunying Liu, and Dahua Lin. Dsnas: Direct neural architecture search without parameter retraining. In *CVPR*, pages 12084–12092, 2020. 2

[23] Shi-Min Hu, Dun Liang, Guo-Ye Yang, Guo-Wei Yang, and Wen-Yang Zhou. Jittor: a novel deep learning framework with meta-operators and unified graph execution. *Information Sciences*, 63(222103):1–21, 2020. 5

[24] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, pages 4700–4708, 2017. 6

[25] Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938. 2, 3

[26] Changlin Li, Jiefeng Peng, Liuchun Yuan, Guangrun Wang, Xiaodan Liang, Liang Lin, and Xiaojun Chang. Block-wisely supervised neural architecture search with knowledge distillation. In *CVPR*, pages 1989–1998, 2020. 2

[27] Guohao Li, Guocheng Qian, Itzel C Delgadillo, Matthias Muller, Ali Thabet, and Bernard Ghanem. Sgas: Sequential greedy architecture search. In *CVPR*, pages 1620–1630, 2020. 2

[28] 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. 3, 4, 5, 7

[29] Xiang Li, Chen Lin, Chuming Li, Ming Sun, Wei Wu, Junjie Yan, and Wanli Ouyang. Improving one-shot nas by suppressing the posterior fading. In *CVPR*, pages 13836–13845, 2020. 2, 5

[30] Yingwei Li, Xiaojie Jin, Jieru Mei, Xiaochen Lian, Linjie Yang, Cihang Xie, Qihang Yu, Yuyin Zhou, Song Bai, and Alan L Yuille. Neural architecture search for lightweight non-local networks. In *CVPR*, pages 10297–10306, 2020. 2

[31] 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, 3

[32] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *CVPR*, pages 82–92, 2019. 2

[33] 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 *ECCV*, pages 19–34, 2018. 6, 7

[34] Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical representations for efficient architecture search. In *ICLR*, 2018. 1, 6

[35] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *ICLR*, 2019. 1, 2, 3, 4, 6, 7

[36] Yun Liu, Yu-Huan Wu, Pei-Song Wen, Yu-Jun Shi, Yu Qiu, and Ming-Ming Cheng. Leveraging instance-, image- and dataset-level information for weakly supervised instance segmentation. *IEEE TPAMI*, 2020. 2

[37] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *ECCV*, pages 116–131, 2018. 7

[38] Asaf Noy, Niv Nayman, Tal Ridnik, Nadav Zamir, Sivan Doveh, Itamar Friedman, Raja Giryes, and Lihi Zelnik. Asap: Architecture search, anneal and prune. In *AISTATS*, pages 493–503, 2020. 3, 4, 6, 7

[39] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019. 5

[40] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In *ICML*, 2018. 1, 2, 6

[41] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *AAAI*, volume 33, pages 4780–4789, 2019. 1, 2, 6, 7

[42] Yao Shu, Wei Wang, and Shaofeng Cai. Understanding architectures learnt by cell-based neural architecture search. In *ICLR*, 2020. 2

[43] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, pages 1–9, 2015. 7

[44] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *CVPR*, pages 2820–2828, 2019. 7

[45] Xiaoxing Wang, Chao Xue, Junchi Yan, Xiaokang Yang, Yonggang Hu, and Kewei Sun. MergeNAS: Merge operations into one for differentiable architecture search. In *IJCAI*, pages 3065–3072, 2020. 6

[46] Yu-Huan Wu, Shang-Hua Gao, Jie Mei, Jun Xu, Deng-Ping Fan, Rong-Guo Zhang, and Ming-Ming Cheng. JCS: An explainable covid-19 diagnosis system by joint classification and segmentation. *IEEE TIP*, 30:3113–3126, 2021. 2

[47] Yu-Huan Wu, Yun Liu, Le Zhang, Wang Gao, and Ming-Ming Cheng. Regularized densely-connected pyramid network for salient instance segmentation. *IEEE TIP*, 2021. 2

[48] Lingxi Xie and Alan Yuille. Genetic cnn. In *CVPR*, pages 1379–1388, 2017. 2

[49] Saining Xie, Alexander Kirillov, Ross Girshick, and Kaiming He. Exploring randomly wired neural networks for image recognition. In *CVPR*, pages 1284–1293, 2019. 2

[50] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. SNAS: stochastic neural architecture search. In *ICLR*, 2019. 1, 2, 3, 4, 6, 7

[51] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. PC-DARTS: Partial channel connections for memory-efficient architecture search. In *ICLR*, 2020. 1, 2, 3, 6, 7

[52] Kaicheng Yu, Christian Sciuto, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. Evaluating the search phase of neural architecture search. In *ICLR*, 2020. 2

[53] Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. Understanding and robustifying differentiable architecture search. In *ICLR*, 2020. 2

[54] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, pages 6848–6856, 2018. 7

[55] Xiawu Zheng, Rongrong Ji, Qiang Wang, Qixiang Ye, Zhenguo Li, Yonghong Tian, and Qi Tian. Rethinking performance estimation in neural architecture search. In *CVPR*, pages 11356–11365, 2020. 2

[56] Dongzhan Zhou, Xinchi Zhou, Wenwei Zhang, Chen Change Loy, Shuai Yi, Xuesen Zhang, and Wanli Ouyang. Econas: Finding proxies for economical neural architecture search. In *CVPR*, pages 11396–11404, 2020. 2

[57] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017. 2

[58] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *CVPR*, pages 8697–8710, 2018. 1, 2, 6, 7