Supplementary of "Fair DARTS: Eliminating Unfair Advantages in Differentiable Architecture Search"

Xiangxiang Chu[1[0000−0003−2548−0605]], Tianbao Zhou[2[0000−0002−2133−059X]], Bo Zhang[1[0000−0003−0564−617X]], and Jixiang Li[1[0000−0001−5949−1498]]

[1] Xiaomi AI Lab
{chuxiangxiang,zhangbo11,lijixiang}@xiaomi.com
[2] Minzu University of China
tianbaochou@163.com

1 Weight-sharing Neural Architecture Search

Weight-sharing in neural architecture search is now prominent because of its efficiency [2, 11, 1, 7]. They can roughly be divided into two categories.

One-stage approaches. Specifically, ENAS [11] adopts a reinforced approach to train a controller to sample subnetworks from the supernet. 'Good' subnetworks yield high rewards so that the final policy of the controller is able to find competitive ones. Notice the controller and subnetworks are trained alternatively. DARTS [10] and many of its variants [16, 17, 6, 5] are also a nested optimization based on weight-sharing but in a differentiable way. Besides, DARTS creates an exclusive competition by selecting only one operation on an edge, opposed to ENAS where more operations can be activated at the same time.

Two-stage approaches. There are some other weight-sharing methods who use the trained supernet as an evaluator [2, 1, 7]. We need to make a distinction here as DARTS is a one-stage approach. The supernet of DARTS is meant to learn towards a single solution, where other paths are less valued (weighted). Instead like in [7], all paths are uniformly sampled, so to give an equal importance on selected paths. As the supernet is used for different purposes, two-stage approaches should be singled out for discussion in this paper.

2 Search Spaces

There are two search spaces extensively adopted in recent NAS approaches. The one DARTS proposed, we denote as S_1, is cell-based with flexible inner connections [10, 5], and the other S_2 is at the block level of the entire network [14, 3, 16]. We use the term *proxy* and *proxyless* to differentiate whether it directly represents the backbone architecture. Our experiments cover both categories, if otherwise explicitly written, the first is searched on CIFAR-10 and the second on ImageNet.

Search Space S_1. Our first search space S_1 (show in Fig. 1) follows DARTS [10] with an exception of excluding the *zero operation*, as done in [18]. Namely,

S_1 works in the previously mentioned DAG of $N = 7$ nodes, first two nodes in cell c_{k-1} are the outputs of last two cells c_{k-1} and c_{k-2}, four intermediate nodes with each has incoming edges from the former nodes. The output node of DAG is a concatenation of all intermediate nodes. Each edge contains 7 candidate operations:

- max_pool_3x3, avg_pool_3x3, skip_connect,
- sep_conv_3x3, sep_conv_5x5,
- dil_conv_3x3, dil_conv_5x5.

Search Space S_2. The second search space S_2 is similar to that of ProxylessNAS [3] which uses MobileNetV2 [12] as its backbone. We make several essential modifications. Specifically, there are $L = 19$ layers and each contains $N = 7$ following choices,

- Inverted bottlenecks with an expansion rate x in (3,6), a kernel size y in (3,5,7), later referred to as ExKy,
- skip connection.

3 Experiment Details

The list of all experiments we perform for this paper are summarized in Table 1.

To summarize, we run DARTS [10] in S_1 and S_2, confirming the aggregation of skip connections in both search spaces. We also show the large discretization gap in S_2.

In comparison, we run Fair DARTS in S_1 and S_2 to show their differences. First, due to our collaborative approach, we can allow a substantial number of skip connections in S_1, see Fig. 7 (supplementary). Second, Fig. 4 (supplementary) exhibit the final heatmap of architectural weights σ where skip connections coexist with other operations, meantime, the values of σ are also close to 0 or 1, which can minimize the discretization gap.

3.1 Search on CIFAR-10 in Search Space S_1

We initialize all architectural weights to 0[3] and set $w_{0-1} = 10$. We comply with the same hyperparameters with some exceptions: a batch size 128, learning rate 0.005, and Adam optimizer with a decay of 3e-3 and beta (0.9, 0.999). Moreover, we comply with the same strategy of grouping training and validation data as [10]. The edge correspondence in search space S_1 is given in Table 2. We also use the first-order optimization to save time.

We illustrate some of the searched cells in Fig. 5 and Fig. 6. The normal cell of FairDARTS-a is rather simple and only 3 nodes are activated, which is previously rarely seen. In the reduction cell, more edges are activated to compensate for the information loss due to down-sampling. As reduction cells are of small proportion, FairDARTS-a thus retains to be lightweight. Other searching results are listed in Table 5.

[3] Its sigmoid output is 0.5 (fair starting point).

3.2 Search on ImageNet in Search Space S_2

We use the SGD optimizer with an initial learning rate 0.02 (cosine decayed by epochs) for the network weights and Adam optimizer with 5e-4 for architectural weights. Besides, we set $\sigma_{threshold} = 0.75$ and $w_{0-1} = 1.0$.

There is a minor issue need to concern. The innate *skip connections* are removed from inverted bottlenecks since skip connections have already been made a parallel choice in our search space.

Table 1. All experiments conducted for this paper. By default, S_1 is for CIFAR-10, S_2 for ImageNet. Note $\mathcal{L}_{val}$ refers to $\mathcal{L}_{val}(w^*(\alpha), \alpha)$. †: Main text. ‡: In the supplementary. $^\star : \mathcal{N}(0, 1)$, run $k = 4$ times, * :$w_{0-1} \in range(16)$

Idx	Method	Search Space	Loss	Optimization	Fig†	Fig‡	Table‡
1	DARTS*	S_1	$\mathcal{L}_{val}$	Bi-level	2,3,4	9,13	
2	DARTS	S_2	$\mathcal{L}_{val}$	Bi-level	1,4		
3	Fair DARTS	S_1	$\mathcal{L}_{val}$	Bi-level	8		
4	Fair DARTS*	S_1	$\mathcal{L}_{val} + w_{0-1}L_{0-1}$	Bi-level	8		
5	Fair DARTS*	S_1	$\mathcal{L}_{val} + 10 * L_{0-1}$	Bi-level	8	3,4,5,6,12,14	5,7
6	Fair DARTS	S_1	$\mathcal{L}_{val} + 10 * L'_{0-1}$	Bi-level		2,10	
7	Fair DARTS	S_1	$\mathcal{L}_{val} + 10 * L_{0-1}$	Single-level		3,11	
8	Fair DARTS	S_2	$\mathcal{L}_{val} + 10 * L_{0-1}$	Bi-level	1,6		
9	DARTS	$S_1 \setminus \{\text{skips}\}$	$\mathcal{L}_{val}$	Bi-level	7	15	
10	Random	$S_1 \cap \{\text{priors}\}$	n/a	n/a			7
11	Random	$S_1 \cap \{\text{priors}\} \cap \{\text{FLOPS} \geq 500M\}$	n/a	n/a			8
12	Noise*	S_1	$\mathcal{L}_{val}$	Bi-level			6
13	DARTS	S_1	$\mathcal{L}_{val} + 10 * L_{0-1}$	Bi-level		8,16	

3.3 Zero-one Loss Comparison

Fig. 2 compares results on two different loss designs. With the proposed loss function L_{0-1} so that Fair DARTS is less subjective to initialization. The sigmoid values reach to their optima more slowly than that of L'_{0-1}. It also gives a fair opportunity near 0.5, the 3×3 dilation convolution on Edge (3,4) first increases and then decreases, which matches the second criteria of loss design.

3.4 Single-level vs. Bi-level Optimization

The 5×5 separable convolution on edge (2, 2) under bi-level setting weighs higher at the early stage but much decreased in the end, which can be viewed as robustness to a local optimum. See Fig. 3.

3.5 Results on COCO Object Detection

We use our models as drop-in replacements of the backbone of RetinaNet [9]. Here we only consider comparable mobile backbones. Particularly, we use the MMDetection [4] toolbox since it provides good implementations for various

Table 2. Edge correspondence in S_1. Edge i is represented as a pair (j, k) where j is an intermediate node, and k is the incoming node. Note j is numbered only on intermediate nodes. And k counts previous cell outputs as well. See Fig. 1 for correspondence

Edge	(j, k)
0	(0, 0)
1	(0, 1)
2	(1, 0)
3	(1, 1)
4	(1, 2)
5	(2, 0)
6	(2, 1)
7	(2, 2)
8	(2, 3)
9	(3, 0)
10	(3, 1)
11	(3, 2)
12	(3, 3)
13	(3, 4)

Table 3. Dropout strategies of DARTS variants

Mothods	Skip Connection	Other Ops
DARTS [10]	No Drop	No Drop
PDARTS [5]	Drop	No Drop
RobustDARTS [18]	Drop	Drop

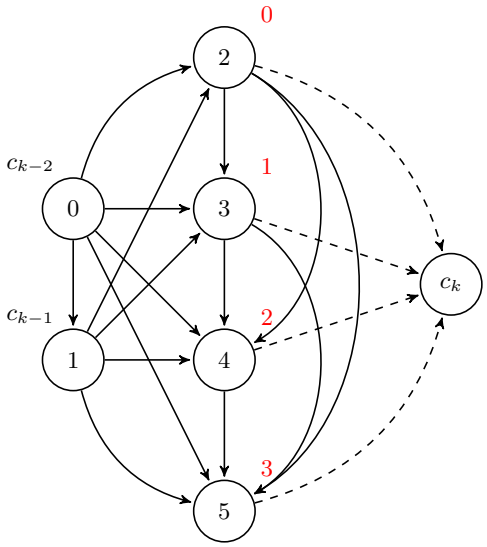


Fig. 1. The DARTS search space at cell-level. Red labels indicate intermediate nodes. The outputs of all intermediate nodes are concatenated to form the output c_k

objective methods. All the models are trained and evaluated on MS COCO dataset (train2017 and val2017 respectively) for 12 epochs with a batch size of 16. The initial learning rate is 0.01 and divided by 10 at epochs 8 and 11. Table 4 shows that our model achieves the best average precision 31.9%.

Table 4. Object detection of various drop-in backbones. †: w/ SE and Swish

Backbones	×+ (M)	Params (M)	Acc (%)	AP (%)	AP_{50} (%)	AP_{75} (%)	AP_S (%)	AP_M (%)	AP_L (%)
MobileNetV2 [12]	300	3.4	72.0	28.3	46.7	29.3	14.8	30.7	38.1
SingPath NAS [13]	365	4.3	75.0	30.7	49.8	32.2	15.4	33.9	41.6
MobileNetV3 [8]	219	5.4	75.2	29.9	49.3	30.8	14.9	33.3	41.1
MnasNet-A2 [14]	340	4.8	75.6	30.5	50.2	32.0	16.6	34.1	41.1
MixNet-M [15]	360	5.0	77.0	31.3	51.7	32.4	17.0	35.0	41.9
FairDARTSC †	386	5.3	77.2	31.9	51.9	33.0	17.4	35.3	43.0

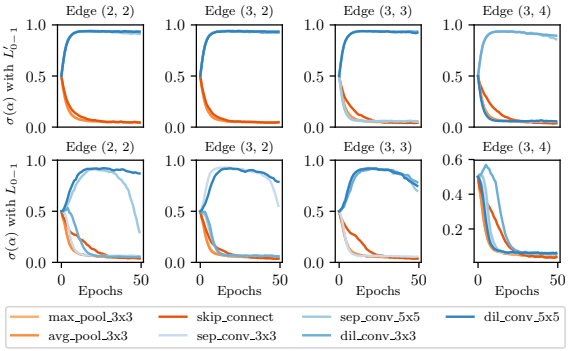


Fig. 2. Comparison of sigmoid evolution when running Fair DARTS with L'_{0-1} (top) and L_{0-1} (bottom) in S_1. With the proposed loss at the bottom, sigmoid values manage to step out of local optima

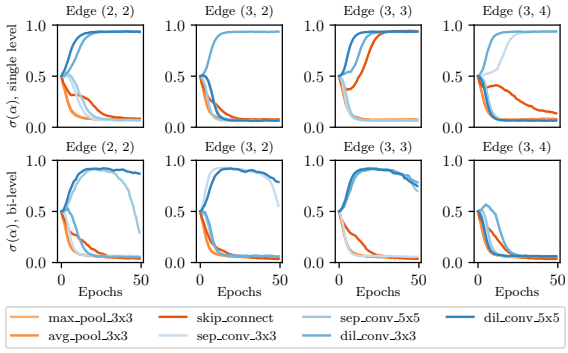


Fig. 3. Comparison of sigmoid evolution when running Fair DARTS with single and bi-level optimization. Bi-level has better robustness to local minimum

4 Figures

4.1 Evolution of Architectural Weights

Fig. 9 and 13 gives the complete *softmax* evolution when running DARTS on CIFAR-10 in S_1. Fig. 15 is a similar case except the skip connection is removed.

Fig. 10 gives the complete *sigmoid* evolution when running Fair DARTS on CIFAR-10 in S_1 with L'_{0-1}.

Fig. 11, 12 gives the complete sigmoid evolution when running Fair DARTS on CIFAR-10 in S_1 with *single-level* and *bi-level* optimization respectively. Fig. 14 is a stacked-bar version of Fig. 12.

4.2 A Special Study: DARTS with L_{0-1}

Although our auxiliary loss L_{0-1} is particularly designed for Fair DARTS, it is interesting to see how DARTS behave under this new condition. Fig. 8 and 16 give us such an illustration, where L_{0-1} has the same weight $w = 10$ as in Fair DARTS. Not surprisingly, under the exclusive competition by softmax, skip connections exploit even more from unfair advantages as we drive the weak-performing operations towards zero (Fig. 16). Noticeably, there is a *domino effect*, as the weakest operation decrease its weight, the second weakest follows, so on and so forth. This effect speeds up the aggregation and as a result, more skip connections stand out (Fig. 8). As the rest better-performing operations are contending with each other, it still finds difficulty to determine which one is the best. Besides, training its inferred best models reaches $96.77 \pm 0.29\%$ on CIFAR-10 (run 7 times each with different seeds), which is not too different from the original DARTS. Therefore, we conclude that applying L_{0-1} alone is not enough to solve the existing problems in DARTS. In fact, L_{0-1} cannot handle the softmax case inherently.

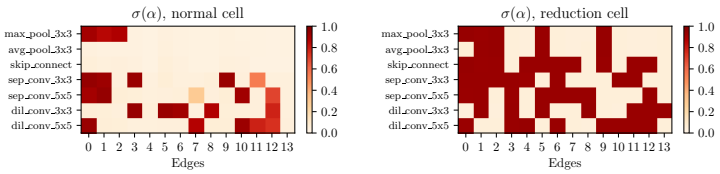


Fig. 4. Heatmap of $\sigma(\alpha)$ in the normal cell and the reduction cell at the last epoch when searching with Fair DARTS on CIFAR-10 (in S_1). As a result of the sigmoid feature and the auxiliary loss L_{0-1}, the values are mainly around 0 and 1

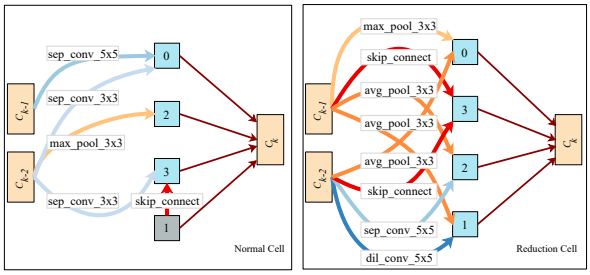


Fig. 5. FairDARTS-a cells searched on CIFAR-10 in S_1

Table 5. Fair DARTS architecture genotypes. See FairDARTS-a in Fig. 5

Model	Architecture Genotype
FairDARTS-b	Genotype(normal=[('sep_conv_3x3', 2, 0), ('sep_conv_3x3', 2, 1), ('sep_conv_3x3', 3, 1), ('dil_conv_3x3', 4, 0), ('sep_conv_5x5', 4, 1), ('dil_conv_5x5', 5, 1)], normal_concat=range(2, 6), reduce=[('skip_connect', 2, 0), ('dil_conv_3x3', 2, 1), ('skip_connect', 3, 0), ('dil_conv_3x3', 3, 1), ('max_pool_3x3', 4, 0), ('sep_conv_3x3', 4, 1), ('skip_connect', 5, 2), ('max_pool_3x3', 5, 0)], reduce_concat=range(2, 6))
FairDARTS-c	Genotype(normal=[('max_pool_3x3', 2, 0), ('sep_conv_5x5', 2, 1), ('dil_conv_3x3', 4, 0), ('dil_conv_5x5', 4, 2), ('skip_connect', 5, 3), ('sep_conv_3x3', 5, 0)], normal_concat=range(2, 6), reduce=[('dil_conv_3x3', 2, 1), ('dil_conv_5x5', 2, 0), ('dil_conv_3x3', 3, 0), ('sep_conv_3x3', 3, 1), ('sep_conv_5x5', 4, 0), ('sep_conv_5x5', 4, 3), ('sep_conv_5x5', 5, 0), ('skip_connect', 5, 1)], reduce_concat=range(2, 6))
FairDARTS-d	Genotype(normal=[('sep_conv_3x3', 2, 0), ('sep_conv_5x5', 2, 1), ('dil_conv_3x3', 3, 1), ('max_pool_3x3', 3, 0), ('dil_conv_3x3', 4, 0), ('dil_conv_3x3', 4, 1), ('sep_conv_3x3', 5, 0), ('dil_conv_5x5', 5, 1)], normal_concat=range(2, 6), reduce=[('max_pool_3x3', 2, 0), ('sep_conv_5x5', 2, 1), ('avg_pool_3x3', 3, 0), ('dil_conv_5x5', 3, 2), ('dil_conv_3x3', 4, 3), ('avg_pool_3x3', 4, 0), ('avg_pool_3x3', 5, 0), ('skip_connect', 5, 3)], reduce_concat=range(2, 6))
FairDARTS-e	Genotype(normal=[('sep_conv_3x3', 2, 0), ('sep_conv_3x3', 2, 1), ('dil_conv_3x3', 4, 1), ('dil_conv_3x3', 4, 2), ('dil_conv_3x3', 5, 0), ('dil_conv_5x5', 5, 1)], normal_concat=range(2, 6), reduce=[('max_pool_3x3', 2, 1), ('max_pool_3x3', 2, 0), ('max_pool_3x3', 3, 1), ('max_pool_3x3', 3, 0), ('sep_conv_5x5', 4, 1), ('max_pool_3x3', 4, 0), ('avg_pool_3x3', 5, 0), ('dil_conv_5x5', 5, 1)], reduce_concat=range(2, 6))
FairDARTS-f	Genotype(normal=[('max_pool_3x3', 2, 0), ('sep_conv_3x3', 2, 1), ('dil_conv_3x3', 3, 1), ('sep_conv_5x5', 4, 1), ('sep_conv_3x3', 5, 0), ('sep_conv_3x3', 5, 1)], normal_concat=range(2, 6), reduce=[('max_pool_3x3', 2, 0), ('max_pool_3x3', 2, 1), ('max_pool_3x3', 3, 0), ('dil_conv_3x3', 3, 1), ('dil_conv_3x3', 4, 2), ('max_pool_3x3', 4, 0), ('max_pool_3x3', 5, 0), ('sep_conv_3x3', 5, 1)], reduce_concat=range(2, 6))
FairDARTS-g	Genotype(normal=[('sep_conv_3x3', 2, 0), ('sep_conv_3x3', 2, 1), ('skip_connect', 4, 3), ('sep_conv_5x5', 4, 1), ('dil_conv_3x3', 5, 0), ('sep_conv_3x3', 5, 1)], normal_concat=range(2, 6), reduce=[('avg_pool_3x3', 2, 1), ('skip_connect', 2, 0), ('skip_connect', 3, 2), ('max_pool_3x3', 3, 1), ('sep_conv_5x5', 4, 3), ('max_pool_3x3', 4, 0), ('dil_conv_3x3', 5, 1), ('dil_conv_3x3', 5, 4)], reduce_concat=range(2, 6))

Table 6. Architecture genotypes when adding Gaussian noise to DARTS. Discussed in Section 6.3 (main text)

Model	Architecture Genotype
0	Genotype(normal=[('sep_conv_3x3', 1), ('sep_conv_3x3', 0), ('dil_conv_5x5', 1), ('dil_conv_3x3', 2), ('sep_conv_3x3', 1), ('dil_conv_5x5', 3), ('dil_conv_5x5', 4), ('dil_conv_3x3', 3)], normal_concat=range(2, 6), reduce=[('max_pool_3x3', 0), ('max_pool_3x3', 1), ('avg_pool_3x3', 0), ('dil_conv_5x5', 2), ('skip_connect', 1), ('max_pool_3x3', 0), ('skip_connect', 3), ('avg_pool_3x3', 1)], reduce_concat=range(2, 6))
1	Genotype(normal=[('dil_conv_3x3', 1), ('sep_conv_3x3', 0), ('dil_conv_3x3', 2), ('dil_conv_3x3', 1), ('sep_conv_3x3', 1), ('dil_conv_5x5', 3), ('dil_conv_3x3', 3), ('dil_conv_5x5', 4)], normal_concat=range(2, 6), reduce=[('dil_conv_5x5', 1), ('skip_connect', 0), ('max_pool_3x3', 0), ('dil_conv_3x3', 2), ('skip_connect', 3), ('skip_connect', 2), ('skip_connect', 2), ('skip_connect', 3)], reduce_concat=range(2, 6))
2	Genotype(normal=[('dil_conv_3x3', 1), ('sep_conv_3x3', 0), ('dil_conv_3x3', 1), ('dil_conv_3x3', 2), ('sep_conv_3x3', 1), ('dil_conv_3x3', 3), ('dil_conv_3x3', 4), ('dil_conv_3x3', 1)], normal_concat=range(2, 6), reduce=[('max_pool_3x3', 0), ('dil_conv_3x3', 1), ('avg_pool_3x3', 0), ('dil_conv_5x5', 2), ('max_pool_3x3', 0), ('skip_connect', 2), ('skip_connect', 2), ('avg_pool_3x3', 0)], reduce_concat=range(2, 6))
3	Genotype(normal=[('dil_conv_3x3', 1), ('skip_connect', 0), ('dil_conv_3x3', 2), ('dil_conv_3x3', 1), ('sep_conv_3x3', 1), ('dil_conv_3x3', 3), ('dil_conv_3x3', 4), ('dil_conv_5x5', 2)], normal_concat=range(2, 6), reduce=[('avg_pool_3x3', 0), ('dil_conv_5x5', 1), ('avg_pool_3x3', 0), ('dil_conv_5x5', 2), ('avg_pool_3x3', 0), ('dil_conv_5x5', 3), ('dil_conv_5x5', 4), ('avg_pool_3x3', 0)], reduce_concat=range(2, 6))

Table 7. Randomly sampled architecture genotypes in S_1 setting $M = 2$. Discussed in Section 6.3 (main text)

Model	Architecture Genotype
0	Genotype(normal=[('skip_connect', 1), ('sep_conv_5x5', 0), ('skip_connect', 1), ('dil_conv_5x5', 0), ('avg_pool_3x3', 1), ('dil_conv_3x3', 0), ('max_pool_3x3', 0), ('sep_conv_3x3', 3)], normal_concat=range(2, 6), reduce=[('dil_conv_3x3', 1), ('dil_conv_5x5', 0), ('max_pool_3x3', 1), ('dil_conv_5x5', 2), ('skip_connect', 0), ('dil_conv_3x3', 1), ('avg_pool_3x3', 4), ('sep_conv_5x5', 1)], reduce_concat=range(2, 6))
1	Genotype(normal=[('skip_connect', 1), ('skip_connect', 0), ('dil_conv_3x3', 0), ('sep_conv_5x5', 2), ('dil_conv_5x5', 1), ('sep_conv_5x5', 3), ('dil_conv_3x3', 3), ('max_pool_3x3', 4)], normal_concat=range(2, 6), reduce=[('sep_conv_3x3', 1), ('sep_conv_3x3', 0), ('dil_conv_3x3', 2), ('max_pool_3x3', 1), ('sep_conv_3x3', 0), ('sep_conv_3x3', 1), ('max_pool_3x3', 2), ('skip_connect', 3)], reduce_concat=range(2, 6))
2	Genotype(normal=[('avg_pool_3x3', 0), ('skip_connect', 1), ('dil_conv_5x5', 0), ('dil_conv_3x3', 2), ('sep_conv_5x5', 2), ('skip_connect', 3), ('avg_pool_3x3', 1), ('avg_pool_3x3', 4)], normal_concat=range(2, 6), reduce=[('avg_pool_3x3', 0), ('avg_pool_3x3', 1), ('avg_pool_3x3', 0), ('sep_conv_5x5', 1), ('sep_conv_3x3', 2), ('avg_pool_3x3', 1), ('sep_conv_3x3', 2), ('max_pool_3x3', 1)], reduce_concat=range(2, 6))
3	Genotype(normal=[('skip_connect', 1), ('dil_conv_3x3', 0), ('sep_conv_3x3', 0), ('avg_pool_3x3', 2), ('dil_conv_5x5', 0), ('dil_conv_3x3', 3), ('max_pool_3x3', 1), ('skip_connect', 4)], normal_concat=range(2, 6), reduce=[('dil_conv_3x3', 0), ('sep_conv_5x5', 1), ('sep_conv_5x5', 1), ('sep_conv_5x5', 0), ('avg_pool_3x3', 0), ('sep_conv_5x5', 2), ('sep_conv_3x3', 0), ('max_pool_3x3', 1)], reduce_concat=range(2, 6))
4	Genotype(normal=[('dil_conv_5x5', 1), ('skip_connect', 0), ('sep_conv_5x5', 0), ('skip_connect', 2), ('sep_conv_5x5', 3), ('dil_conv_5x5', 2), ('avg_pool_3x3', 0), ('max_pool_3x3', 3)], normal_concat=range(2, 6), reduce=[('dil_conv_3x3', 0), ('dil_conv_3x3', 1), ('max_pool_3x3', 0), ('avg_pool_3x3', 2), ('max_pool_3x3', 0), ('avg_pool_3x3', 2), ('dil_conv_3x3', 0), ('dil_conv_3x3', 2)], reduce_concat=range(2, 6))
5	Genotype(normal=[('sep_conv_5x5', 0), ('skip_connect', 1), ('sep_conv_3x3', 2), ('sep_conv_3x3', 0), ('avg_pool_3x3', 2), ('skip_connect', 0), ('sep_conv_3x3', 0), ('dil_conv_5x5', 3)], normal_concat=range(2, 6), reduce=[('avg_pool_3x3', 1), ('sep_conv_5x5', 0), ('dil_conv_3x3', 2), ('skip_connect', 1), ('avg_pool_3x3', 2), ('skip_connect', 1), ('sep_conv_3x3', 4), ('dil_conv_3x3', 1)], reduce_concat=range(2, 6))
6	Genotype(normal=[('skip_connect', 1), ('sep_conv_5x5', 0), ('max_pool_3x3', 0), ('dil_conv_3x3', 1), ('sep_conv_5x5', 1), ('avg_pool_3x3', 0), ('skip_connect', 4), ('sep_conv_5x5', 0)], normal_concat=range(2, 6), reduce=[('dil_conv_5x5', 1), ('sep_conv_3x3', 0), ('sep_conv_5x5', 0), ('dil_conv_5x5', 1), ('max_pool_3x3', 1), ('max_pool_3x3', 3), ('dil_conv_5x5', 2), ('max_pool_3x3', 1)], reduce_concat=range(2, 6))

Table 8. Randomly sampled architecture genotypes in S_1 setting $M = 2$ and multiply-adds > 500M. Discussed in Section 6.3 (main text)

Model	Architecture Genotype
0	Genotype(normal=[('skip_connect', 1), ('sep_conv_3x3', 0), ('skip_connect', 0), ('dil_conv_3x3', 2), ('sep_conv_3x3', 1), ('dil_conv_5x5', 2), ('sep_conv_3x3', 2), ('dil_conv_5x5', 1)], normal_concat=range(2, 6), reduce=[('skip_connect', 0), ('sep_conv_5x5', 1), ('avg_pool_3x3', 2), ('dil_conv_5x5', 0), ('sep_conv_3x3', 0), ('dil_conv_5x5', 2), ('sep_conv_5x5', 1), ('dil_conv_5x5', 2)], reduce_concat=range(2, 6))
1	Genotype(normal=[('dil_conv_3x3', 0), ('sep_conv_3x3', 1), ('skip_connect', 2), ('skip_connect', 1), ('sep_conv_5x5', 0), ('dil_conv_3x3', 1), ('sep_conv_5x5', 1), ('dil_conv_5x5', 4)], normal_concat=range(2, 6), reduce=[('dil_conv_3x3', 0), ('dil_conv_3x3', 1), ('avg_pool_3x3', 1), ('skip_connect', 0), ('dil_conv_5x5', 1), ('skip_connect', 3), ('skip_connect', 0), ('max_pool_3x3', 3)], reduce_concat=range(2, 6))
2	Genotype(normal=[('sep_conv_5x5', 0), ('max_pool_3x3', 1), ('sep_conv_5x5', 1), ('skip_connect', 0), ('skip_connect', 2), ('max_pool_3x3', 1), ('sep_conv_5x5', 3), ('sep_conv_3x3', 2)], normal_concat=range(2, 6), reduce=[('sep_conv_5x5', 0), ('skip_connect', 1), ('max_pool_3x3', 1), ('sep_conv_5x5', 2), ('dil_conv_5x5', 3), ('max_pool_3x3', 0), ('dil_conv_3x3', 1), ('max_pool_3x3', 2)], reduce_concat=range(2, 6))
3	Genotype(normal=[('sep_conv_3x3', 0), ('max_pool_3x3', 1), ('dil_conv_5x5', 2), ('dil_conv_5x5', 0), ('sep_conv_5x5', 3), ('sep_conv_5x5', 0), ('skip_connect', 3), ('skip_connect', 0)], normal_concat=range(2, 6), reduce=[('avg_pool_3x3', 0), ('sep_conv_5x5', 1), ('avg_pool_3x3', 1), ('dil_conv_3x3', 0), ('dil_conv_5x5', 3), ('sep_conv_5x5', 2), ('avg_pool_3x3', 1), ('dil_conv_5x5', 4)], reduce_concat=range(2, 6))
4	Genotype(normal=[('dil_conv_5x5', 0), ('sep_conv_5x5', 1), ('sep_conv_3x3', 2), ('skip_connect', 0), ('sep_conv_5x5', 3), ('sep_conv_5x5', 0), ('avg_pool_3x3', 1), ('skip_connect', 0)], normal_concat=range(2, 6), reduce=[('max_pool_3x3', 1), ('skip_connect', 0), ('dil_conv_5x5', 1), ('dil_conv_5x5', 2), ('sep_conv_5x5', 0), ('sep_conv_3x3', 1), ('avg_pool_3x3', 1), ('skip_connect', 0)], reduce_concat=range(2, 6))
5	Genotype(normal=[('sep_conv_5x5', 1), ('sep_conv_5x5', 0), ('skip_connect', 2), ('sep_conv_3x3', 0), ('dil_conv_5x5', 2), ('dil_conv_3x3', 3), ('max_pool_3x3', 0), ('skip_connect', 2)], normal_concat=range(2, 6), reduce=[('max_pool_3x3', 0), ('dil_conv_5x5', 1), ('max_pool_3x3', 2), ('skip_connect', 0), ('dil_conv_5x5', 0), ('sep_conv_5x5', 3), ('sep_conv_3x3', 1), ('dil_conv_5x5', 3)], reduce_concat=range(2, 6))
6	Genotype(normal=[('avg_pool_3x3', 0), ('sep_conv_5x5', 1), ('sep_conv_5x5', 0), ('skip_connect', 1), ('dil_conv_5x5', 0), ('sep_conv_5x5', 1), ('skip_connect', 4), ('sep_conv_5x5', 0)], normal_concat=range(2, 6), reduce=[('dil_conv_5x5', 1), ('avg_pool_3x3', 0), ('avg_pool_3x3', 2), ('sep_conv_5x5', 0), ('max_pool_3x3', 0), ('dil_conv_5x5', 2), ('sep_conv_5x5', 3), ('skip_connect', 0)], reduce_concat=range(2, 6))

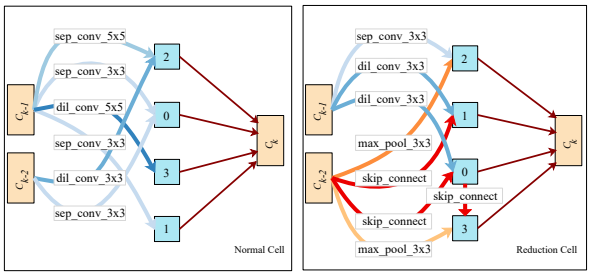


Fig. 6. FairDARTS-b cells searched on CIFAR-10 in S_1

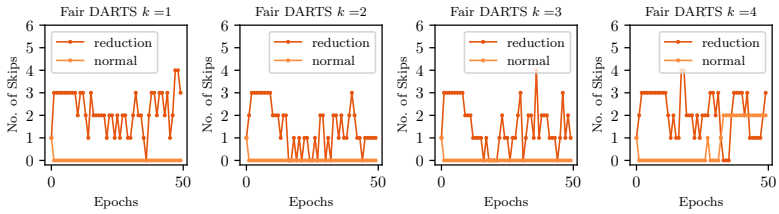


Fig. 7. The number of dominant skip connections when searching with Fair DARTS on CIFAR-10. Here we select top-2 operations as dominants as in DARTS (our proposed way selects dominants according to $\sigma_{threshold}$), our method can still escape from too many skip connections. Note that we choose the first 4 experiments to show

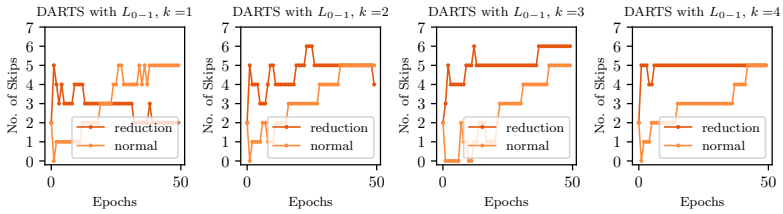
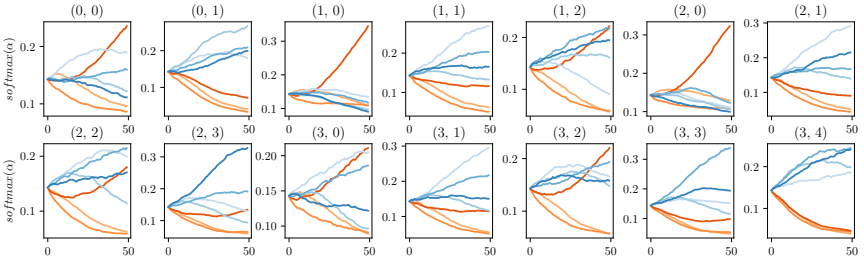
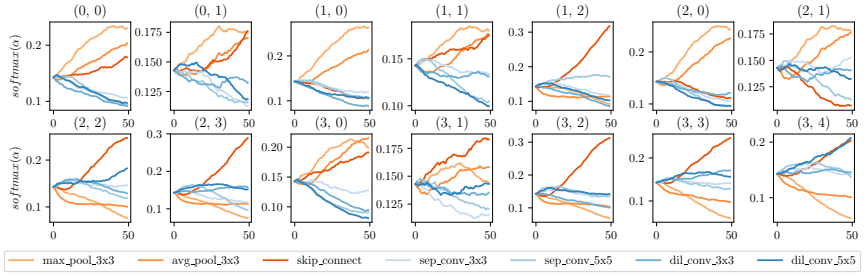


Fig. 8. The number of dominant skip connections when searching with DARTS equipped with L_{0-1} on CIFAR-10. Notice the aggregation of skip connections gets even worse

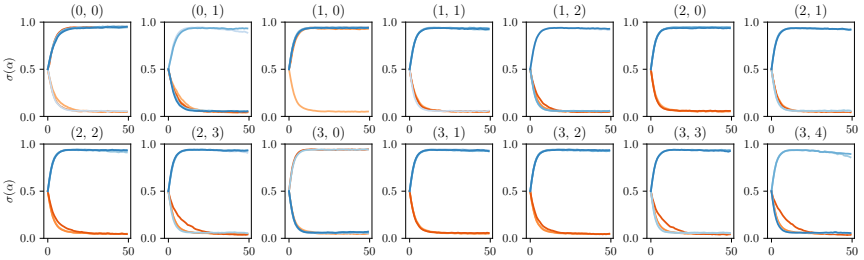


(a) Normal Cell

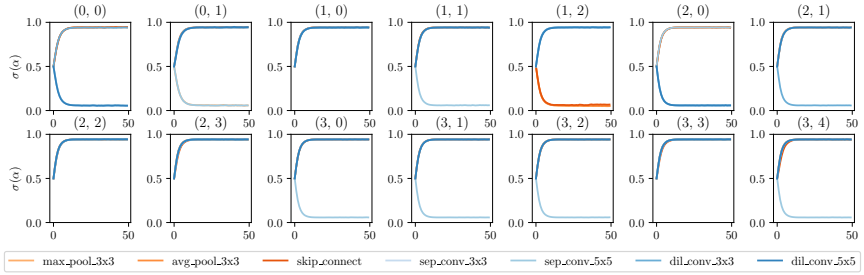


(b) Reduction Cell

Fig. 9. The softmax evolution when running DARTS on CIFAR-10 in S_1 ($k = 3$)

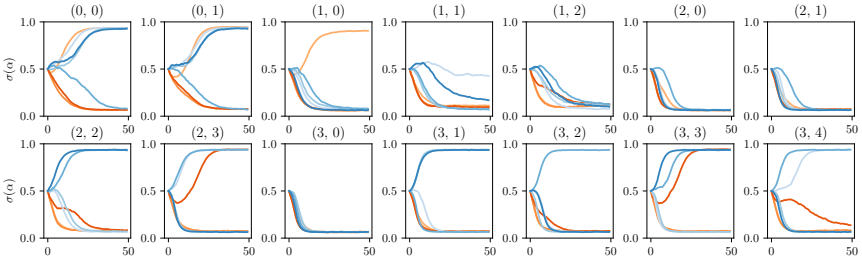


(a) Normal Cell

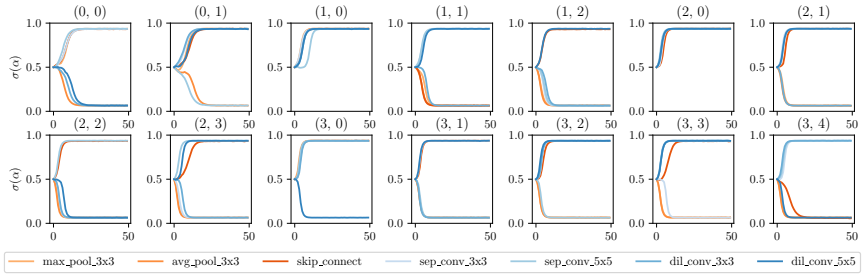


(b) Reduction Cell

Fig. 10. The sigmoid evolution when running Fair DARTS with L'_{0-1} loss in S_1

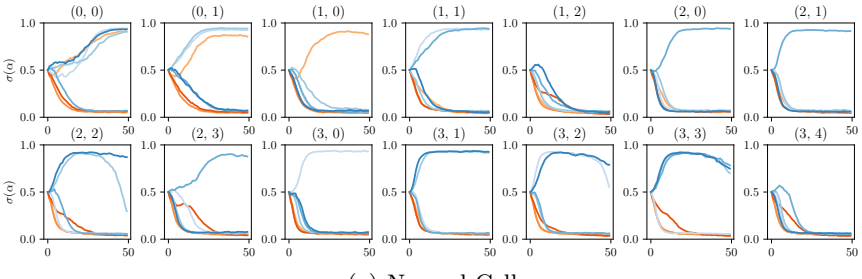


(a) Normal Cell

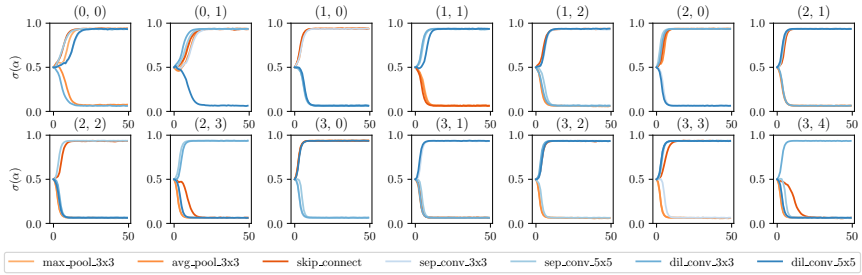


(b) Reduction Cell

Fig. 11. The sigmoid evolution when running Fair DARTS with single-level optimization in S_1

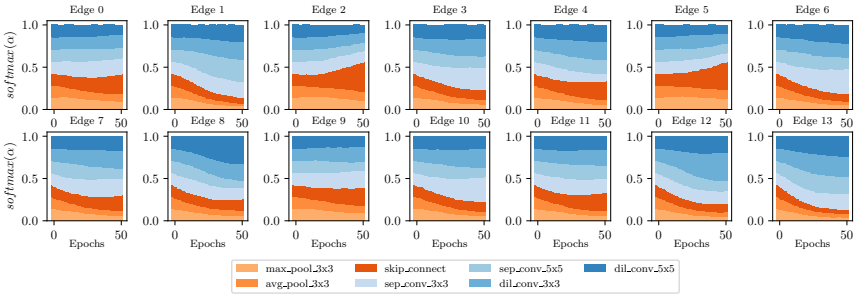


(a) Normal Cell

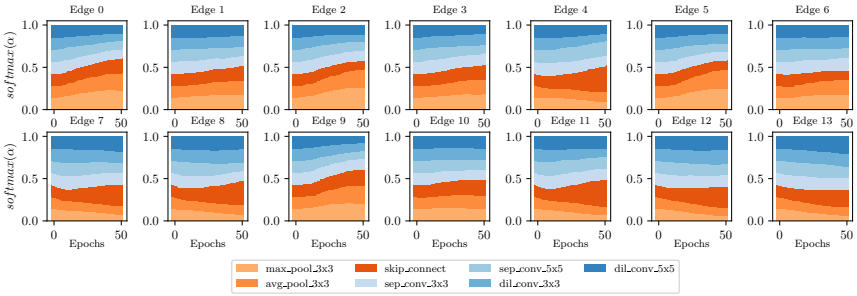


(b) Reduction Cell

Fig. 12. The sigmoid evolution when running Fair DARTS with bi-level optimization in S_1

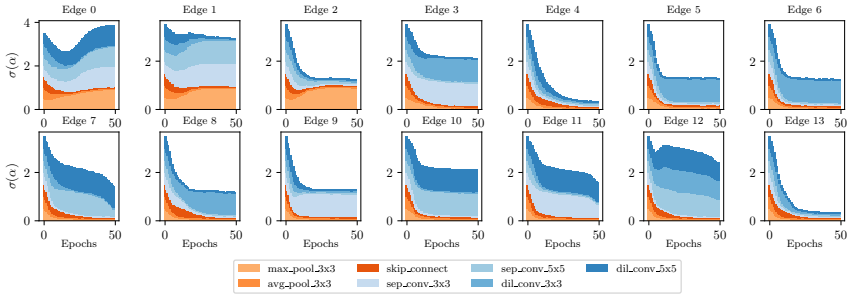


(a) Normal Cell

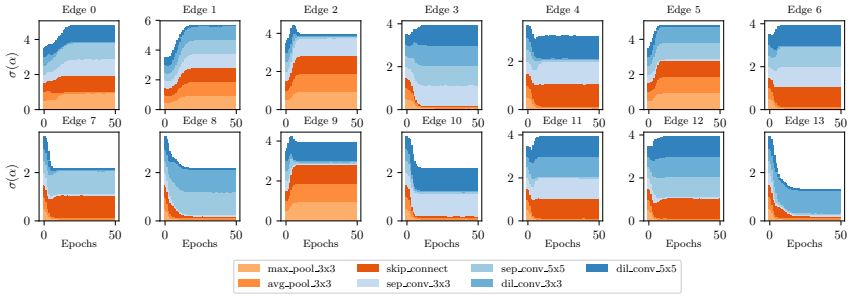


(b) Reduction Cell

Fig. 13. The evolution of $softmax(\alpha)$ when running DARTS on CIFAR-10 in S_1. Skip connections on edge 0,2,4,5,11 in the normal cell and edge 4,7,8,11,12 in the reduction cell gradually suppress others caused by unfair advantage

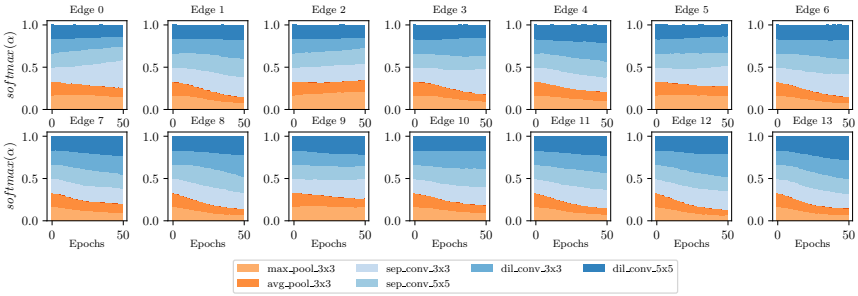


(a) Normal Cell

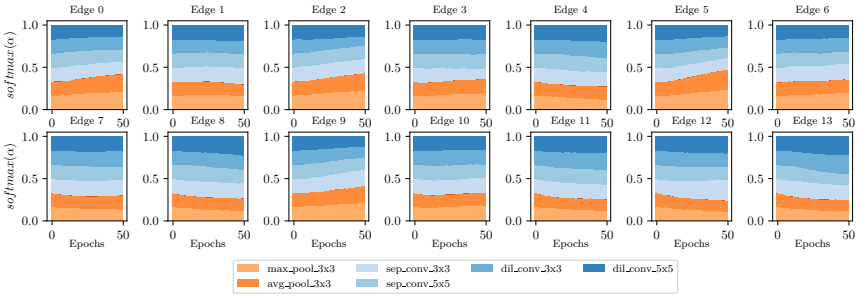


(b) Reduction Cell

Fig. 14. The evolution of $\sigma(\alpha)$ when running Fair DARTS on CIFAR-10 in S_1. Skip connections enjoy an equal opportunity under collaborative competition. See edge 1,8,12 in normal cell and almost all edges in reduction cell

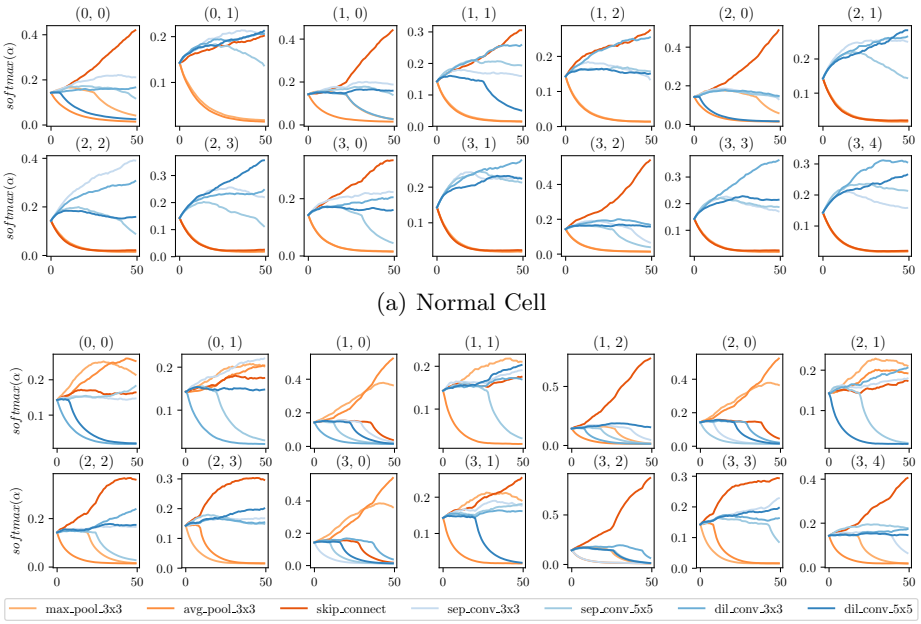


(a) Normal Cell



(b) Reduction Cell

Fig. 15. The evolution of $\sigma(\alpha)$ when running DARTS on CIFAR-10 in S_1 without skip connections. With unfair advantages removed, all operations are encouraged to demonstrate each real strength



(a) Normal Cell

(b) Reduction Cell

Fig. 16. The evolution of $softmax(\alpha)$ when running DARTS with the auxiliary loss L_{0-1} enabled on CIFAR-10 in S_1

References

1. Bender, G., Kindermans, P.J., Zoph, B., Vasudevan, V., Le, Q.: Understanding and Simplifying One-Shot Architecture Search. In: International Conference on Machine Learning. pp. 549–558 (2018)
2. Brock, A., Lim, T., Ritchie, J.M., Weston, N.: SMASH: One-Shot Model Architecture Search through HyperNetworks. In: International Conference on Learning Representations (2018)
3. Cai, H., Zhu, L., Han, S.: ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware. In: International Conference on Learning Representations (2019)
4. Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., et al.: Mmdetection: Open mmlab detection toolbox and benchmark. arXiv preprint arXiv:1906.07155 (2019)
5. Chen, X., Xie, L., Wu, J., Tian, Q.: Progressive Differentiable Architecture Search: Bridging the Depth Gap between Search and Evaluation. International Conference on Computer Vision (2019)
6. Dong, X., Yang, Y.: Searching for a Robust Neural Architecture in Four GPU Hours. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1761–1770 (2019)
7. Guo, Z., Zhang, X., Mu, H., Heng, W., Liu, Z., Wei, Y., Sun, J.: Single Path One-Shot Neural Architecture Search with Uniform Sampling. In: European Conference on Computer Vision (2020)
8. Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al.: Searching for MobileNetV3. In: International Conference on Computer Vision (2019)
9. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: International Conference on Computer Vision (2017)
10. Liu, H., Simonyan, K., Yang, Y.: DARTS: Differentiable Architecture Search. In: International Conference on Learning Representations (2019)
11. Pham, H., Guan, M.Y., Zoph, B., Le, Q.V., Dean, J.: Efficient Neural Architecture Search via Parameter Sharing. In: International Conference on Machine Learning (2018)
12. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: MobileNetV2: Inverted Residuals and Linear Bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4510–4520 (2018)
13. Stamoulis, D., Ding, R., Wang, D., Lymberopoulos, D., Priyantha, B., Liu, J., Marculescu, D.: Single-Path NAS: Designing Hardware-Efficient ConvNets in less than 4 Hours. In: European Conference on Machine Learning and Knowledge Discovery in Databases (2019)
14. Tan, M., Chen, B., Pang, R., Vasudevan, V., Le, Q.V.: MnasNet: Platform-Aware Neural Architecture Search for Mobile. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2019)
15. Tan, M., Le., Q.V.: MixConv: Mixed Depthwise Convolutional Kernels. The British Machine Vision Conference (2019)
16. Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y., Keutzer, K.: FBNet: Hardware-Aware Efficient ConvNet Design via Differentiable Neural Architecture Search. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2019)

17. Xie, S., Zheng, H., Liu, C., Lin, L.: SNAS: Stochastic Neural Architecture Search. International Conference on Learning Representations (2019)
18. Zela, A., Elsken, T., Saikia, T., Marrakchi, Y., Brox, T., Hutter, F.: Understanding and robustifying differentiable architecture search. In: International Conference on Learning Representations (2020), https://openreview.net/forum?id=H1gDNyrKDS