Rethink DARTS Search Space and Renovate a New Benchmark

Jiuling Zhang [1 2] **Zhiming Ding** [2 1]

Abstract

DARTS search space (DSS) has become a canonical benchmark for NAS whereas some emerging works pointed out the issue of narrow accuracy range and claimed it would hurt the method ranking. We observe some recent studies already suffer from this issue that overshadows the meaning of scores. In this work, we first propose and orchestrate a suite of improvements to frame a larger and harder DSS, termed LHD, while retaining high efficiency in search. We step forward to renovate a LHD-based new benchmark, taking care of both discernibility and accessibility. Specifically, we re-implement twelve baselines and evaluate them across twelve conditions by combining two underexpolored influential factors: transductive robustness and discretization policy, to reasonably construct a benchmark upon multi-condition evaluation. Considering that the tabular benchmarks are always insufficient to adequately evaluate the methods of neural architecture search (NAS), our work can serve as a crucial basis for the future progress of NAS. https://github.com/chaoji90/LHD

Table 1. Scores of the benchmark on DSS.

METHODS	CIFAR-10		IMAGENET-1K
	ERROR (%)	*#param* (M)	
MILENAS (HE ET AL., 2020)	2.51±0.11	3.87	24.7
PC-DARTS (XU ET AL., 2020)	2.57±0.07	3.6	25.1
GAEA-ERM (LI ET AL., 2021)	2.50±0.06	3.7	24.3
DRNAS (CHEN ET AL., 2021B)	2.54±0.03	4.0	24.2
GIBBSNAS (XUE ET AL., 2021)	2.53±0.02	4.1	24.6
SP-DARTS (ZHANG & DING, 2021)	2.50±0.07	3.5	24.4
DARTS- (CHU ET AL., 2021)	2.59±0.08	3.5	24.8
β-DARTS (YE ET AL., 2022)	2.53±0.08	3.83	24.2

1. Introduction

DARTS relaxes categorical selection through a convex combination of the architecture parameters and operation outputs. In the search phase, architecture parameters α and operation weights ω are alternately optimized on validation set and training set respectively through a bilevel optimization objective. Henceforth, we collectively refer to the line of works explicitly parameterize architecture search by relaxing the categorical operation selection to a differentiable operation distribution as DARTS and specify the method pioneered by Liu et al. (2019) as vanilla DARTS. We also use the name of search space to refer to the benchmark on that space when the context is unambiguous. Research community has established a benchmark surrounding DSS which has been extensively used to evaluate NAS methods (Mehta et al., 2022). Given a current benchmark, two desiderata are discernibility and accessibility. Li & Talwalkar (2020) studied the indeterministic training of the methods and demonstrated that empirically, validation accuracy (*val_acc*) fluctuates over multiple trials, sometimes exceeding 0.1%, for the same finalnet (search result) under the same seed. Accordingly, we refer to the case where the accuracy margin in rank is less than 0.1% as **n**arrow **r**ange **r**anking (NRR). Table 1 illustrates current scores on DSS where the average **a**ccuracy **m**argin between **a**djacent items of the method **r**anking (AMAR) is only 0.012% on CIFAR-10 which is 8.3× smaller than 0.1%. Some studies (Yang et al., 2020; Garg et al., 2020; Yu et al., 2020b; Wang et al., 2020) have pointed out that the narrow accuracy range of DSS causes baselines indiscernible and impairs the validity of the benchmark (more related works in Appx.A).

Yang et al. (2020) systematically studied the evaluation phase on DSS and incrementally quantified the contribution of different modules (Auxiliary Towers, Drop Path, Cutout, Channels, AutoAugment, Epochs) to the final scores. They emphasized that introducing new tricks in *evaluation* has a much greater impact on performance than employing different NAS methods. By contrast to their work focused on the manifest influential factors, we observe more subtleties, i.e. several minute deviations of the evaluation protocol are introduced by succeeding studies, including different drop-path rate, learning rate decay target, batch size, seed, minor revise of operations (batch normalization after poolings). These minutiae are partially inherited by subsequent researches (Xue et al., 2021; Li et al., 2021; Zhang & Ding,

[1] University of Chinese Academy of Sciences, Beijing, China [2] Institute of Software, Chinese Academy of Sciences, Beijing, China. Correspondence to: Zhiming Ding <zhiming@iscas.ac.cn, zhangjiuling19@mails.ucas.edu.cn>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

2021; Chen et al., 2021b) but are intractable unless carefully investigating every released code. We combine the above minutiae and re-evaluate the finalnet (search result) reported by vanilla DARTS and obtain $+0.14\%$ (2.76→2.62) improvement which is much larger than 0.012% and gives us reason to believe that the cumulative effect of these modifications must be non-trivial in light of the NRR of Table 1.

To sum up, **margin** in rank is critical for the confidence of a benchmark. For this, we find some previous studies use *t-test* to measure the confidence of the score comparisons (Hooker et al., 2019; Yu et al., 2020b; Pourchot et al., 2020). So in this paper, we utilize both AMAR and the average *t-test* (Welch, 1947) margin between adjacent items of ranking (TMAR) as the measurements of discernibility. For a list of methods L, we can i) Sort L in terms of accuracies (rank); ii) Get pair-wise margins (accuracy gap for AMAR, *t-test* for TMAR) of adjacent items of the sorted L; iii) Average all margins to get AMAR or TMAR. AMAR measures the absolute margin of accuracies and TMAR takes both accuracy and variance into account when examining the *t-test* value. We can also see from the 4*th* column in Table 1 that the narrow range is likely to be an intrinsic feature of the search space and will not be rectified by simply evaluating on more challenging data. This observation is also verified on other datasets by Yang et al. (2020).

In general, a good space of NAS is expected to exclude human bias and be flexible enough to cover a wide variety of candidates (He et al., 2021). Most of the previous search spaces were proposed as the *byproducts of methods*. To challenge the art scores, these studies always have an incentive to introduce as many artifacts as possible to make their space easier to traverse so that the methods are less error-prone. However, *this design motive is diametrically opposite to the discernible objective of benchmark thereby leads to the problem of NRR we are observing now*. On the contrary, AutoML-Zero (Real et al., 2020) specifically pointed out that the accuracy of a large enough search space should be sparse which is very the critical character of a discernible benchmark. Even further, too many artifacts also cause the search space easy to be overfitted. He et al. (2020) observed that the parameter scale (*#param*) is closely related to *val_acc* and outperforms art zero-shot estimators on DSS (Ning et al., 2021). FLOPs and *#param* remain highly correlated and exhibits consistent correlation with *val_acc* on both DSS and our LHD, so in this paper we focus on inspecting *#param*. We believe that the approach to get better score on benchmark by just looking for larger capacity operations is definitely not our expectation for NAS. The space of a benchmark should be both **large** and **difficult**, so that the methods are not prone to attain higher scores by opportunistically overfitting the space. In this work, we propose some improvements to overhaul DSS and formulate a **l**arger and **h**arder new **D**SS, namely LHD.

Based on LHD, we step ahead to newly construct a multi-condition evaluation benchmark in which we focus on combining the evaluation of both transductive robustness and discretization policies. The 'transductive' here refers to search and expect to find the optimal architecture in-situ on the search dataset. The benefits of the multi-condition evaluation is three-fold: i) Further enhance discriminability, even if some methods perform close under a single condition, we can compare them by taking all conditions into account; ii) Make benchmark more challenging, claiming superior across multiple conditions is much harder than on a single condition; iii) Uncover many more methodological characteristics and preferences that are unobservable within current counterpart that solely provides a few scores. Our contributions can be summarized as follows:

1. We propose i) Node aggregation enhancement: input-softmax; ii) New searchable blocks: searchable polynary operations, searchable cell outputs of sum and concatenation; iii) Primitive refinement: unified convolution primitive. We orchestrate all to construct a new search space that is demonstrably larger and harder than DSS. Through this work, we succeed in weakening the correlation between *#param* and *val_acc* from 0.52 (KD τ) on DSS (Ning et al., 2021; Yang et al., 2020) to 0.29/0.26/0.20 on three valid spaces of LHD.

2. We renovate a new benchmark on LHD involves assessing the transductive robustness of **twelve** baselines over four discretization policies across three datasets. No single method outperforms others under all conditions and the overall ranks are rather unstable across conditions both of which demonstrate that our benchmark is more challenging for methods to show generalizability and claim superior.

2. New Search Space

Preliminary: DSS formulates a cell-based search space with N nodes $X = \{x^n | n \in [1..N]\}$ where $[1..N] := [1, N]$ and E compound edges $G = \{g^e_{i,j} | e \in [1..E], i \in [1, j-1], j \in (n_i, N]\}$ where n_i refers to the number of cell inputs. Compound edge $g_{i,j}$ connects node i to j and associates three attributes: candidate operation set $O_{i,j} = \{o^m_{i,j} | m \in [1..M]\}$, corresponding operation parameter set $A_{i,j} = \{\alpha^m_{i,j} | m \in [1..M]\}$, probability distribution of the parameters $\boldsymbol{a}_{i,j} = \mathrm{softmax}(A_{i,j})$. Every intermediate node is connected to all its predecessor through an edge $g_{i,j}(x_i) = \langle \boldsymbol{a}_{i,j}, O_{i,j}(x_i) \rangle$ where $i < j$. Typically, a unified set of operation candidates $O = \{o^m | m \in [1..M]\}$ is defined for all edges. Each edge subsumes all operation candidates to express the transformations between nodes. Every node aggregates the outputs of all incoming edges into the new feature maps. The network that encodes all architecture candidates is called supernet. We use primitive to describe an indivisible substructure of operations and de-

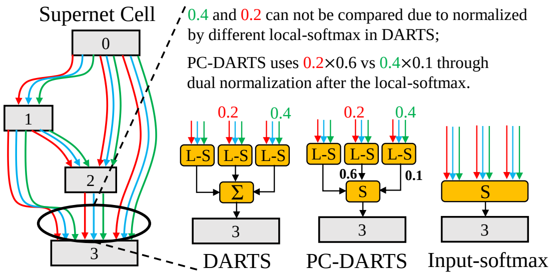

Supernet Cell
0
1
2
3
0.4 and 0.2 can not be compared due to normalized by different local-softmax in DARTS;
PC-DARTS uses 0.2×0.6 vs 0.4×0.1 through dual normalization after the local-softmax.
0.2
0.4
L-S
Σ
DARTS
0.6
0.1
S
PC-DARTS
Input-softmax

Figure 1. Zoom in on the input end of the node 3 to illustrate the differences. L-S denotes the local-softmax in DARTS.

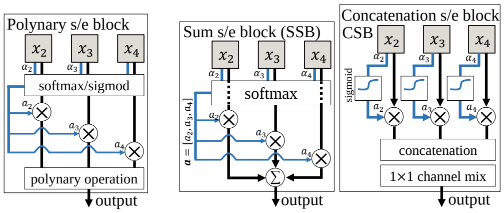


Figure 2. Polynary s/e block on the left and its two instances on the right. One parameter attach to each path to express the significance that can be optimized through SGD. Parameters are first squashed by softmax or sigmoid and then weight the feature maps passed through the paths. LHD uses s/e blocks on the right as cell outputs.

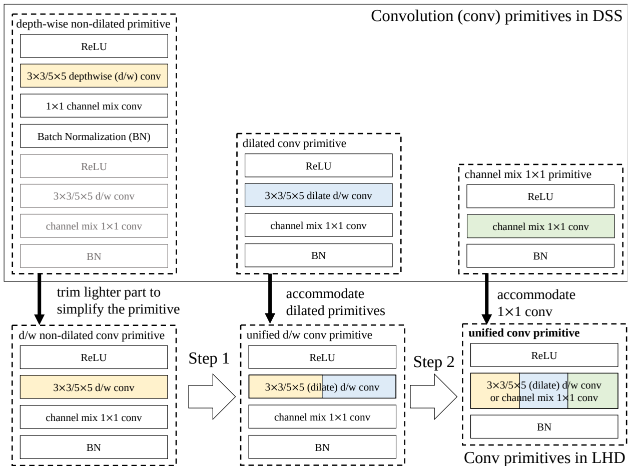


Figure 3. Refinement and unification aim to simplify and unify the structure of convolution primitives to trim redundant artifacts, the operations are still conducted separately not merged (Wang et al., 2021c).

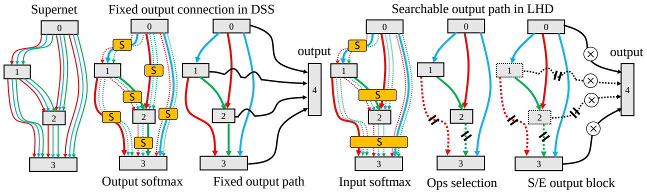


Figure 4. Fixed output path in DSS on the left compared to s/e output block and removable intermediate nodes in LHD on the right. By orchestrating input-softmax and polynary s/e block, the intermediate node on the rightmost can be removed from the finalnet (search result) in the following two cases: i) Node is neither selected by the cell output nor selected by any subsequent nodes like node 2; ii) Node is neither selected by the cell output nor any its succeeding node is selected by the output like node 1.

note block as a substructure containing nodes, edges, paths which can be searchable (s/e) or unsearchable (u/e). We use rounded and solid rectangle, dashed box to represent cell, block and primitive respectively.

Design principle: We first propose a suite of improvements and new searchable blocks. For clarity, we separately delineate their motivations, problem solved, and solutions. We then orchestrate them all to frame the new LHD. An ablation of these improvements are provided in Section 3. Finally, we give a number of features of the new search space. For the overall design principle, *we enlarge search space and trim artifacts while keeping its empirical memory cost roughly the same*. Artifacts refer to the unsearchable structures in macro design and the over-designed primitives of operation candidates, both of which introduce human bias, limit the flexibility of space, make the space easy to traverse and less error prone. We realize beforehand that some similar ideas were proposed and examined separately (Wu et al., 2021b; Jiang et al., 2019) but not collectively. All these researches are far from framing a generally applicable new space to replace DSS. We formulate LHD in Appx.B.

2.1. New modules to frame LHD

Input-softmax: *Motivation: DSS suffers from the limit of sub-graphs due to the absence of a mechanism to compare the significance of operations across edges*. As shown in Figure 1, the softmax is applied on each compound edge without considering their connection pattern in graph, namely local-softmax by I-DARTS (Jiang et al., 2019). Node aggregation in DSS is simply sums up all the feature maps of incoming edges as $x_j = \sum_{i<j} g_{i,j}(x_i)$. PC-DARTS partially solves this by employing path normalization $p_{i,j} g_{i,j}$ to double normalize the significances from different edges depicted in Figure 1. *Solution: We address this limitation by placing softmax directly before aggregation on the node input end* instead of edges to simultaneously normalize elements across all incoming edges shown on the rightmost of Figure 1. This way, the significance of any operation $o^m_{i,j}$ toward node j can be fairly compared through the value of $a^m_{i,j}$ for all the combinations of $m \in [1, M]$ and $i \in [1, j-1]$.

Search Polynary Operation: *Motivation: Innovative use of the polynary operations is often a key improvement in handcrafted regime*, e.g. addition in ResNet (He et al., 2016) and concatenation in DenseNet (Huang et al., 2017). However, parameters only attach on unary operations (single input single output) on each edge in DSS. *Solution: Generaliz-*

ing the merit of DARTS to search polynary operations is straightforward by associating parameter with each path to express its significance that can be optimized through gradient descent. Figure 2 conceptually visualizes the s/e block of polynary operation on the left. The path parameters are first squashed and then weight and aggregate the feature maps over all paths.

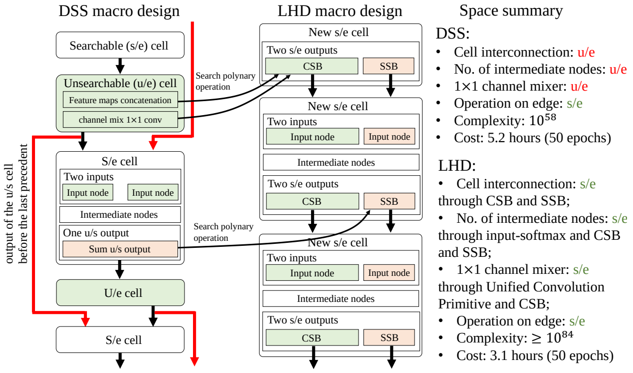
Unified Convolution Primitive: *Motivation: As shown in Figure 3, the convolution primitives are designed to be rather complicated and the non-dilated primitives are deliberately deeper than the dilated counterparts in DSS*. For trimming artifacts to reduce human bias, *solution: we first unify the dilated and non-dilated depth-wise (d/w) primitives specified as step 1 to form the unified d/w primitive. We step forward to incorporate the d/w primitive and 1×1 channel mixer into an unified structure of the primitive*. The unified primitive ultimately accommodates all convolution candidates and ensures their similar structures in the space.

Orchestration to Build LHD: *Motivation: DARTS is incapable to search none (zero) operation directly. So all intermediate nodes are densely connected to the cell output and unsearchably attend in finalnet as shown on the left of Figure 4*. The valid size of space is thus severely restricted to solely depends on the number of edges M because the search on DSS is limited to only the operation selection inside edges without considering their interconnection topology. *Solution: We relax the connection path between intermediate nodes and cell output and let methods search the inter-cell connection pattern through optimizing the path significance in the search phase*. Our goal is to make the intermediate nodes **removable** thereby decouple the finalnet from the design of supernet in Figure 4.

Motivation: Artifacts of the u/e cell and fixed skip connection in the macro design of DSS as shown on left of Figure 5. As a design choice, we regard the path of SSB as an selection with exclusivity in contrast to the non-exclusive path selection of CSB. So we normalize the output path of CSB and SSB by gating and softmax respectively as detailed on the right of Figure 2. *Solution: Detailed in the caption of Figure 5*. By pruning the fixed inter-cell skip connections highlighted in red in Figure 5, LHD expects NAS method to learn the appropriate gradient path by themselves in the search phase such that the methods cannot obtain better score by simply choosing larger capacity architecture.

2.2. Characteristics of LHD

Case study: SSB normalizes the output path by softmax allude to that at least the strongest path will be reserved. Meanwhile, the sigmoid gating in CSB have a potential to close all path that leads to finalnet reduces to a single-input single-output structure as shown in Figure 6a. The cell output is a feature maps aggregation by summation that coincides with the building block of ResNet in Figure 6c. In another case shown by Figure 6b, SSB selects only one output path from the intermediate node that reads feature maps only from the cell input through a skip connection. The cell input and output are straight through without transformation that of course won't bring any reasonable performance but will be a meaningful failure case. NAS methods can also find cell with two parallel branches as shown by Figure 6d and the representations are thus learned independently.



Space summary

DSS:
- Cell interconnection: u/e
- No. of intermediate nodes: u/e
- 1×1 channel mixer: u/e
- Operation on edge: s/e
- Complexity: 10^{58}
- Cost: 5.2 hours (50 epochs)

LHD:
- Cell interconnection: s/e through CSB and SSB;
- No. of intermediate nodes: s/e through input-softmax and CSB and SSB;
- 1×1 channel mixer: s/e through Unified Convolution Primitive and CSB;
- Operation on edge: s/e
- Complexity: $\geq 10^{84}$
- Cost: 3.1 hours (50 epochs)

Figure 5. Macro design of DSS involves two major artifacts: i) u/e cell for concatenation and channel mix; ii) fixed skip connections between cells. We trim artifact 'i' by instantiating a polynary s/e block as a concatenation s/e block (CSB) and use CSB as an output of s/e cell to replace the u/e cell in DSS as shown by the black arrow. We then trim 'ii' but retain dual inputs of the s/e cell and instantiate another sum s/e block (SSB) as the second output to match the number of inputs of s/e cell as shown on the right.

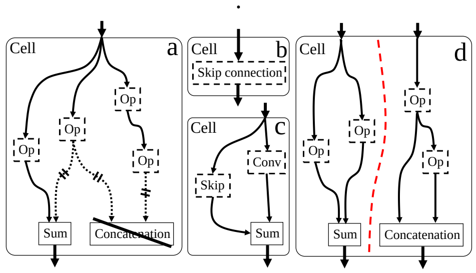


Figure 6. Three corner cases (a,b,d) accommodated by LHD but absent in DSS shows the versatility and inclusivity of LHD.

Computation and memory overhead: We strike a balance between the space augmentation and the search acceleration. For vanilla DARTS, primitive refinement reduces the size of superent by 70% (1.93M→0.56M) and gives rise to a memory surplus to increase intermediate nodes from four to five. Furthermore, replacing the u/s cell in DSS with the CSB allows us to increase the batch size of the search phase by 15% (152→176). On the whole, the depth of the supernet is reduced by two-thirds and the time overhead of the search phase of vanilla DARTS on CIFAR-10 is 40% lower than that on DSS (5.2h→3.1h on RTX 3090, like-for-like comparison after aligning all other conditions).

Table 2. Outline specific characteristics of baselines in the benchmark. Penultimate column lists the benchmarks used for evaluation in original papers. NB201 is NAS-BENCH-201 (Dong & Yang, 2020), NB1S1 is NAS-BENCH-1Shot1 (Zela et al., 2020), S1∼S4 are proposed by (Arber Zela et al., 2020). (T) denotes the tabular benchmark. PC denotes partial channels. SP denotes sparse $\boldsymbol{\alpha}$ distribution and OS denotes operation shortcut. GAEA-B refers to GAEA-Bilevel and GAEA-E refers to GAEA-ERM (Li et al., 2021). β-DARTS added an additional term in loss to regularize architecture parameters.

Baseline	Optimization		Relaxation	Gradient	SP	PC	OS	Evaluations adopted	Codebase
DARTS (Liu et al., 2019)	bilevel	joint	softmax	normal	no	no	no	DSS	quark0/darts
MiLeNAS (He et al., 2020)	**mixlevel**	joint	softmax	normal	no	no	no	DSS	chaoyanghe/MiLeNAS
DrNAS (Chen et al., 2021b)	bilevel	joint	**dirichlet**	normal	no	no	no	DSS, NB201 (T)	xiangning-chen/DrNAS
GAEA-B (Li et al., 2021)	bilevel	joint	softmax	**exponentiated**	no	no	no	DSS, NB201 (T)	liamcli/gaea_release
GAEA-E (Li et al., 2021)	**silevel**	joint	softmax	**exponentiated**	no	no	no	DSS, NB201 (T), NB1S1 (T)	liamcli/gaea_release
GDAS (Dong & Yang, 2019)	bilevel	**sampling**	**gumble-softmax**	normal	**yes**	no	no	DSS, NB201 (T)	D-X-Y/AutoDL-Projects
SP-DARTS (Zhang & Ding, 2021)	bilevel	joint	**low-temp softmax**	normal	**yes**	no	no	DSS, NB201 (T), S1∼S4	chaoji90/SP-DARTS
PC-DARTS (Xu et al., 2020)	bilevel	joint	softmax	normal	no	**yes**	no	DSS	yuhuixu1993/PC-DARTS
SurgeNAS (Luo et al., 2022)	**silevel**	joint	softmax	normal	no	no	**yes**	NB201 (T)	-
DARTS- (Chu et al., 2021)	bilevel	joint	softmax	normal	no	no	**yes**	DSS, NB201 (T), S1∼S4	Meituan-AutoML/DARTS-
β-DARTS (Ye et al., 2022)	bilevel	joint	softmax	normal	no	no	no	DSS, NB201 (T)	Sunshine-Ye/Beta-DARTS

Table 3. Comparisons of discretization policies.

Search space	Discretization policy	Operation selection	Output path selection	Complexity
DSS	original	top-2	fixed, unsearchable	10^{18}
LHD	Base	top-2	**threshold control**	10^{31}
	1M	as Base	as Base	as Base
	3ops	**top-3**	as Base	10^{41}
	4out	as Base	**top-4**	10^{28}

Complexity of the continuous DAGs: We increase the complexity from 10^{58} of DSS to $\geq 10^{84}$ of LHD. Analysis is detailed in Appx.C. The valid subspace is determined by the discretization policy actually used in Table 3.

3. LHD Benchmark

Appropriate benchmark grounds existing method and inspires further research. Our work is not to frame a new space and try every existing method to bring us a good architecture. To some extent, we actually embarrass existing methods by removing their dependent artifacts, enlarging search space, searching upon different conditions. Nonetheless, we will show the potentiality of some search results in the last part. We assess twelve baselines over four discretization policies across three standard benchmark datasets tot twelve organized conditions to validate the transductive robustness and exhibit the impact of the discretization policies on ranking. Baselines are curated with specific characteristics that are highlighted in Table 2 from which we can see that DSS is almost compulsory for the adequate evaluation of NAS methods. We do our best to fully understand the codebases released on DSS and migrate them to LHD.

Transductive robustness in search: NAS aims to automate the general network design. Robustness of the search phase is critical since tuning hyperparameters on each space for each dataset is prohibitive and unsustainable. We can first reasonably assume that all baselines' settings have been specially tuned on DSS on CIFAR-10 (DSS&C10). We expect to achieve reasonable performance by directly applying these settings on LHD&C10. After that, we transfer these settings of the search phase to LHD&CIFAR-100 (C100) and LHD&SVHN to evaluate the transductive robustness across datasets (model settings are detailed in Appx.D). In our benchmark, we adopt the latest and most intuitive search protocol from (Li et al., 2021; Dong & Yang, 2020; He et al., 2020; Xue et al., 2021) which can be summarized as follows: i) Uniformly sample n seeds from 1 to 100,000; ii) Search n times with the seeds independently; iii) Evaluate n results separately and take the average *val_acc*. Our benchmark set n to five, the maximum value of previous studies, greater than three trials in (Dong & Yang, 2020; Ying et al., 2019) and four trials in (He et al., 2020; Xue et al., 2021).

Discretization policies: We propose four discretization policies in Table 3. **Base** closely follows the top-2 operation selection in DSS while **3ops** acts as a straightforward alternative to select top-3 operations on each input-softmax. For the s/e cell outputs, Base thresholds SSB by 0.2 (starting point of the five s/e paths) and CSB by the mean gating level of all paths. **4out** is a variant of this by simply selecting top-4 out of five paths for both SSB and CSB so that the cells are densely interconnected. We discuss the tuning-based path selection method (Wang et al., 2021b) in detail in Appx.F.

DARTS expects the gradient-based optimization to select the appropriate operation. This selection subsequently affects the network scale as shown in Table 1 but leaves the question of whether the performance gaps come from different *#param* rather than architectural merit. (Tay et al., 2022) also identifies that models operate well in one scale does not guarantee its performant in another. In practice, networks are widely scaled by increasing depth and width. We therefore come up policy **1M** that adopts the same architecture as Base, but first scale it up by increasing the stacked cells from 20 to 25 and then align *#param* through augmenting *init_channels* until the finalnet attains 1.5M *#param*.

Complexity of the valid search space: Original policy of

Table 4. We report mean and standard deviation of *val_acc* as the main scores. We also report the averaged *#param* (M) to uncover the preference of baselines in the perspective of model capacity. We report additional top-1 and top-3 scores because some methods are trapped in rare failure cases. Evaluation results on C10 are shown here, results on C100 and SVHN are provided in Appx.E.

C10	Base			1M			3ops			4out		
Method	*val_acc* (%)	*#param*	top-1/top3	*val_acc* (%)	*#param*	top-1/top3	*val_acc* (%)	*#param*	top-1/top3	*val_acc* (%)	*#param*	top-1/top3
DARTS	93.58±4.11	0.67	96.40/96.02	93.89±4.42	1.54	96.92/96.41	92.24±4.01	0.74	95.53/94.49	91.73±4.82	0.90	96.54/94.44
DrNAS	94.14±2.03	0.57	95.97/95.28	94.48±2.20	1.55	96.39/95.77	94.52±1.12	0.64	95.63/95.12	92.75±3.80	0.86	94.83/94.72
GAEA-B	**95.93±0.46**	0.68	96.52/96.18	**96.20±0.59**	1.56	96.73/96.56	**95.70±0.83**	0.77	96.40/96.19	**96.17±0.43**	0.96	96.64/96.40
GAEA-E	94.74±0.56	0.91	95.25/95.08	94.88±0.48	1.60	95.47/95.22	94.97±0.53	1.13	95.32/95.25	94.89±0.44	1.03	95.25/95.14
GDAS	94.90±0.22	0.55	95.24/95.04	95.62±0.28	1.53	96.28/95.79	95.24±0.42	0.61	95.58/95.53	**95.92±0.31**	0.98	96.28/96.12
MiLeNAS	**95.60±0.43**	0.66	96.11/95.82	**95.99±0.55**	1.53	96.86/96.30	**95.46±0.81**	0.78	96.31/95.94	**95.67±0.37**	0.98	96.21/95.91
PC-DARTS	95.11±0.52	0.71	95.86/95.42	95.45±0.56	1.56	96.22/95.76	95.42±0.52	0.83	96.04/95.77	95.40±0.51	1.09	96.09/95.73
Random	95.09±0.69	0.64	95.75/95.51	95.58±0.63	1.56	96.34/95.98	95.60±0.11	0.77	95.75/95.68	95.48±0.29	0.92	95.79/95.67
SP-DARTS	**95.83±0.39**	0.64	96.22/96.08	**96.12±0.52**	1.55	96.84/96.43	**95.53±0.78**	0.76	96.22/96.07	94.10±2.18	0.94	96.01/95.40
DARTS-	92.84±2.77	0.66	96.48/94.56	93.14±2.72	1.54	96.98/94.70	93.57±1.91	0.75	96.28/94.76	91.69±2.52	0.84	95.85/93.01
β–DARTS	95.01±0.75	0.60	95.84/95.26	95.16±0.98	1.52	96.03/95.47	93.80±1.14	0.71	94.95/94.19	92.11±4.47	0.92	95.22/94.26
SurgeNAS	94.38±0.99	0.78	95.07/94.95	94.66±1.01	1.55	95.60/95.36	94.77±1.24	0.96	95.40/95.33	94.64±0.56	0.98	95.27/94.97

Table 5. Discernibility measurements of the ranks under different conditions. Larger values of AMAR and TMAR in LHD imply greater margins between items in rank, thereafter yield more discernible ranking to alleviate NRR in DSS.

Condition	C10&DSS	C10&Base	C10&1M	C10&3ops	C10&4out	C100&Base	C100&1M	C100&3ops	C100&4out	SVHN&Base	SVHN&1M	SVHN&3ops	SVHN&4out
AMAR/_top3 (%)	0.012/0.005	0.29/0.17	0.28/0.10	0.32/0.09	0.41/0.25	1.60/0.74	2.18/1.03	1.12/1.12	2.45/1.01	0.27/0.017	0.27/0.027	0.20/0.078	0.31/0.114
TMAR/_top3	0.30/0.08	0.50/0.64	0.43/0.30	0.43/0.24	0.67/1.07	0.78/0.24	0.60/0.43	0.72/0.45	1.12/0.77	0.79/0.18	0.98/0.25	0.96/1.30	1.18/1.47

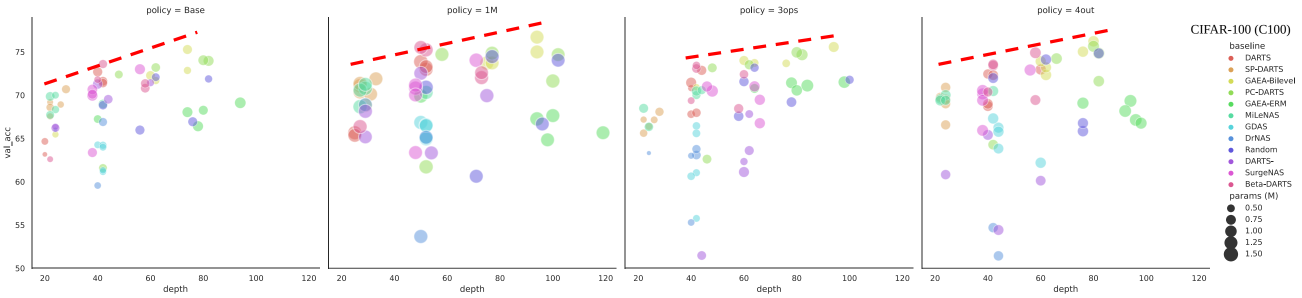


Figure 7. Distributions of the search results on the coordinate frame of *val_acc* versus depth. We specifically put the results on C100 here because the performance ceiling is noticeably positively correlated with the finalnet (search result) depth, which is not obvious on C10 and SVHN (see Appx.E). Methods prefer deeper or shallower structures thereby struggle to achieve consistent scores across conditions.

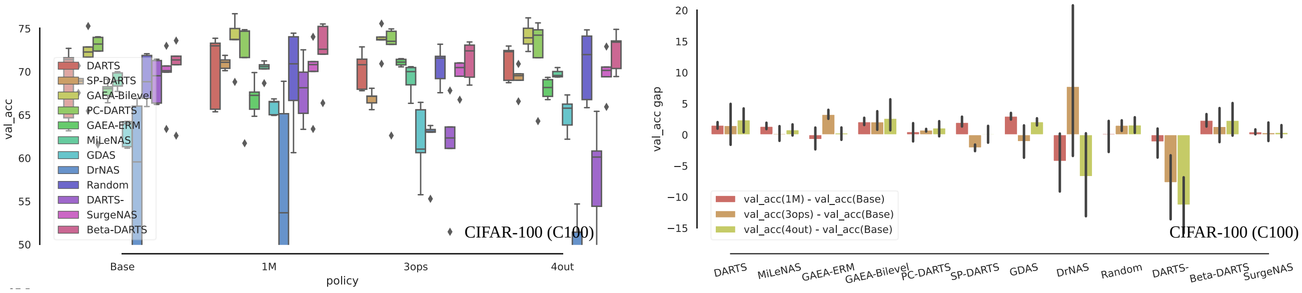


Figure 8. (left) Compare baseline *val_acc*s on four policies respectively along x axis. Some whiskers are truncated for the clarity in the main scope. (right) Exhibits the differences in *val_acc*s between Base and other policies to illustrate the effect of the discretization on different methods. Each bar contains the scores of five trials. Results on C10 and SVHN and are provided in Appx.E due to space limit.

DSS allows $\prod_{k=1}^{4} C_{k+1}^{2} \times 7^2 = \prod_{k=1}^{4} \frac{(k+1)k}{2} \times 7^2 \approx 10^9$ possible valid cell after discretization and the total complexity of the normal and reduction searches is approximate 10^{18}. Possible discretized valid cell in LHD can be obtained through $((\sum_{i=1}^{N-1} C_N^i)^2 \prod_{j=2}^{N+1} C_{Mj}^k)^2$ with M operation candidates and N intermediate nodes. k is the specified top-k operation selection on each input-softmax.

Contribution ablation: From the complexity of 10^{18} on DSS, contribution of each improvement can be ablated: i) For N=4 and fixed path single cell output, input-softmax enlarges valid search space by one order of magnitude to 10^{19}; ii) Combining dual cell s/e outputs with input-softmax to yield removable intermediate node augments another six orders of magnitude from step 'i' to the valid search space 10^{25}; iii) Increasing intermediate nodes from four to five accounts for another six orders of magnitude to the final

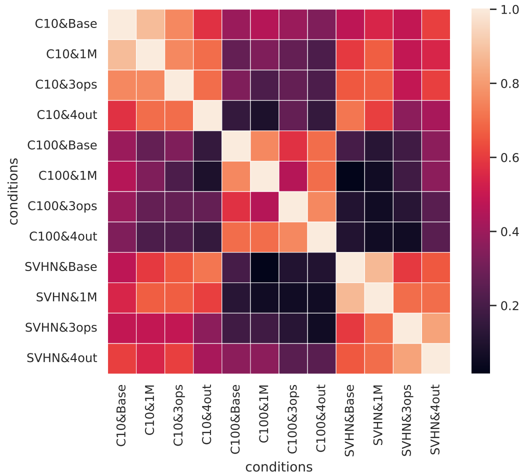


Figure 9. Small KD corresponds to low rank correlation. Method performs well on one condition does not guarantee its precedence on others. Single condition evaluation to claim superior could be misleading and not generalizable.

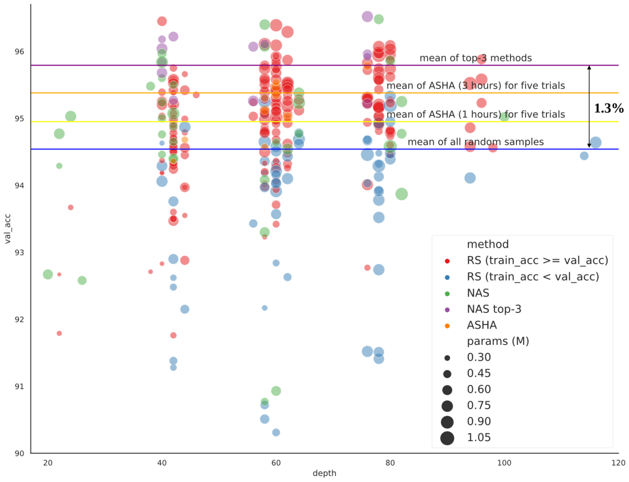


Figure 10. Random samples (RS) and random search (ASHA five trials) on C10&Base. ASHA has been proven to be an art partial training method that outperforms leading adaptive search strategies for hyperparameter optimization (Li et al., 2020). Results on C10&3ops, C10&4out and more details are provided in Appx.I.

valid complexity 10^{31} of Base as shown in Table 3.

Evaluation protocol: To refrain the additional cost added by multi-condition evaluation, we choose a relatively lower *#param* regime ($\leq$1M or 1.5M) than DSS ($\geq$3.5M). We employ seed 0 which is the same as DSS for all evaluations on LHD and propose a i-value based heuristic regularization protocol (more in Appx.G) to tackle the diversity of search results. Nevertheless, we believe that our benchmark is friendly and accessible to any practitioner since the entire pipeline can be delivered on a single GPU.

3.1. Results of the Benchmark

The main scores are reported in Table 4. Figure 7 illustrates the distribution of the search results in terms of depth, *val_acc* and *#param*. Figure 8 shows the results grouped by policies on the left and shows the performance differences between policies on the right. Kendall's Tau (KD τ) is widely used to study the rank correlations on NAS (Yu et al., 2020b; Park et al., 2020; Zhang et al., 2020). Figure 9 illustrates a heatmap of KD to show a pairwise correlation of the twelve baselines' ranking over twelve conditions. We also demonstrate the improvements of discernibility in Table 5.

Observations from the results (1$\sim$5): **1**. If a method prefers a deep and large cell, such as GAEA-ERM and PC-DARTS, it is often more difficult to learn the proper gradient path that deteriorates performance while scaling up from Base to 1M; **2**. In contrast, if a method prefers simple and shallow architecture, such as SP-DARTS and GDAS, it is likely to fail to yield good performance on the conditions prefer deeper structures, e.g. on C100; **3**. It's hard to balance the preferences at the same time, e.g. SP-DARTS is one of the top performant art on both C10 and SVHN but is poor on C100. PC-DARTS is just the opposite that works well on C100. But this dilemma is our very intention in designing LHD and balancing the contradictory is also our expectation of the superior method; **4**. Both transductive robustness and discretizations have a significant impact on methods ranking as illustrated in Figure 9; **5**. We notice that even with the improved discernibility of the space, there are still local indiscernible in ranks, e.g. AMARs of the top-3 methods are ≤ 0.1 under some conditions in Table 5. Multi-condition evaluations compensate this that if a method claims superior, it should prove that across most conditions (if not all) rather than upon a single condition with marginal score differences. We specify more observations on the characteristics of methods in Appx.H.

3.2. Random Sampling and Random Search

We conduct further studies on LHD and observe (1$\sim$5): **1**. (Yang et al., 2020) evaluated 200+ samples on DSS and observed "all within a range of 1% after a standard full training on C10" i.e. narrow accuracy range. We evaluate 250+ random samples on C10&Base which exhibit a much larger accuracy gap in Figure 10; **2**. More than 25% of the random samples have *train_acc*<*val_acc* after full 600 epochs of training which also occurs in some NAS methods that favor deep and large cell, both of which imply an absence of appropriate gradient path and highlight the deliberate harder part of LHD. **3**. The correlation between *#param* and *val_acc* is 0.29/0.26/0.20 for random samples on BASE/3ops/4out on C10 as opposed to 0.52 on C10&DSS (Ning et al., 2021), demonstrating that LHD breaks the tight correlation (He et al., 2020) under even lower *#param* regime. LHD thereby

Table 6. Evaluate different settings. RA denotes RandAugment (Cubuk et al., 2020) and CM denotes CutMix (Yun et al., 2019). i controls the drop-path rate which is also positively correlated with the cells' interconnection density.

Dataset	Architecture	Evaluations		
		#param (M)	Settings	*val_acc* (%)
C10	SpC10-B	0.49	i=0.04, RA+CM	96.79
	Rs3ops	0.8/2.5/3.5	i=0.06/0.07/0.08	96.84/97.51/97.70
		2.5/3.5	i=0.05/0.06, RA	97.69/97.82
SVHN	SpSVHN-B	0.58	different recipe	96.64

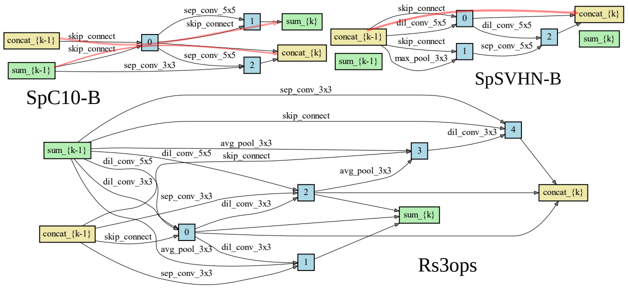


Figure 11. High performant normal cells with simple connection patterns searched by SP-DARTS on C10&Base and SVHN&Base, named SpC10-B and SpSVHN-B respectively. Learned gradient paths are evident and highlighted in red. The Rs3ops is much more complex that comes from random sampling on C10&3ops.

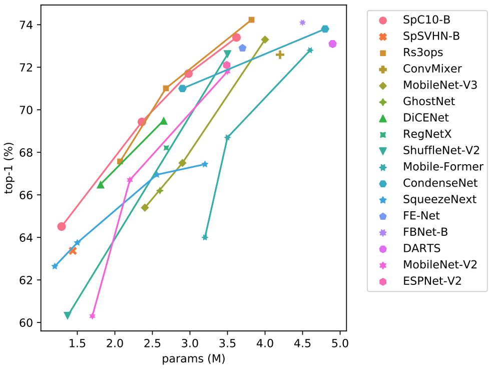


Figure 12. Comparisons on ImageNet-1k under mobile regime.

makes method more difficult to obtain high scores through overfitting *#param* and the validity of the architecture per se is more important; **4**. As marked by Figure 10, the mean accuracy of the top-3 methods outperforms random sampling by a large margin (1.3%) on C10&Base in contrast to many previous studies pointing to only a trivial gap (<0.5%) between art NAS methods and random sampling on C10&DSS (Yu et al., 2020c; Yang et al., 2020; Garg et al., 2020; Lindauer & Hutter, 2020), in particular underpins the discernable improvement in proposal. **5**. Random search is conducted by combining ASHA which is previously studied on DSS (Li & Talwalkar, 2020) and shows competitive results. The NAS method only needs one-shot 3-hour training to obtain search results for all policies (Base/3ops/4out) whereas ASHA must be applied separately, so we provide both 1-hour and 3-hour results for ASHA in Figure 10.

4. Beyond the Benchmark

We visualize three high performant normal cells in Figure 11. Both SpC10-B and SpSVHN-B retain only three out of five intermediate nodes in the space due to removable search in proposal. Furthermore, method can obtain one input/output structure like SpSVHN-B that underscores the effectiveness of the s/e output blocks in finding inter-cell connection patterns. Additionally, We can easily recognize the learned gradient paths in Figure 11 that replace the fixed inter-cell skip connections in DSS. Beside the simple cells, LHD is also capable to encode rather complicated connection patterns as shown by Rs3ops on the bottom pane.

We try to ablate the evaluation settings and ultimately deliver a strong scores 96.79% on C10 with only 0.49M *#param* in Table 6. For comparison, ConvMixer (Trockman & Kolter, 2022) equipped with RA+CM+Mixup+Random Erasing attains the best 95.19% (0.594M) and 95.88% (0.707M) in their ablations. Besides, Rs3ops yields on par or superior *val_acc* with 30% lesser *#param* comparing with the optimal results on DSS in Table 1. We also test another drastically different recipe where DenseNet40_k12 delivers 96.22% with similar *#param* but 2× more MACs under the same recipe on SVHN[1]. To evaluate the transferability on ImageNet (224×224), we closely follow the evaluation recipe on DSS that moderately increase the capacity to accommodate higher resolution inputs. Our scores are competitive against a wide range of baselines shown in Figure 12.

5. Compare with Queryable Benchmark

Tabluar benchmark has an unquestionable efficiency upside. However, its downside also comes from the efficiency that makes methods easier to be tuned and overfit. We have witnessed a fair number of studies claimed to stably achieve the global optimal on NB201 (Dong & Yang, 2020), but none of these methods yield that level of superior on DSS or LHD; The recent NAS-Bench (NB) study (Mehta et al., 2022) underscores the similar point in studying non-one-shot methods on more benchmarks. Mehta et al. (2022) even appealed to stop focusing much on smaller NB201 (Dong & Yang, 2020) and NB101 (Zela et al., 2020; Ying et al., 2019) and rather embrace larger and novel new NBs. One-shot methods can be applied on four NBs in (Mehta et al., 2022) and three of which, including NB201 and NB101, are tabular benchmarks in addition to DSS. Since evaluating NAS only on tabular benchmarks is always considered inadequate, our work is one crucial complement rather than an exclusivity to the counterparts, in particular for one-shot methods and topology search. On the other hand, tabular benchmarks hardly provide effective insight for other related fields due to their limited size of spaces. In contrast,

[1] longrootchen/densenet-svhn-classification-pytorch

more practical DSS has inspired follow-up researches in various forms (Shu et al., 2020; Han et al., 2022; Knyazev et al., 2021). Surrogate benchmark (Siems et al., 2020) is another type of queryable NB that predicts space statistics by pre-training tens of thousands of architectures. Surrogating twelve conditions in our case requires pre-training about one million samples which it's currently unrealistic.

Acknowledgements

This work is supported by the National Key R & D Program of China (2022YFF0503900)

6. Conclusion

In this paper, we dig into hardening and enlarging the canonical benchmark space DSS under limited resources, taking care of both discernibility and accessibility. We conduct a comparative study to establish a multi-condition evaluation benchmark and focus on comparing the unique contribution of each method but leaving their possible combinations for the future work. In particular, we provide abundant art baselines and all the scores can be used out of box without laborious repetition. For fair comparison, we strongly recommend practitioners to only tune on one condition and transfer the exact settings to others. The results after tuned can be provided separately if necessary. We believe that the benefits of our study are multifaceted as we provide a basis for the further research, including a versatile and inclusive search space, a more revealing and all accessible benchmark and the research progress of the fair comparison of methods.

References

Arber Zela, T. E., Saikia, T., Marrakchi, Y., Brox, T., and Hutter, F. Understanding and robustifying differentiable architecture search. In *International Conference on Learning Representations*, volume 3, pp. 7, 2020.

Chen, M., Wu, K., Ni, B., Peng, H., Liu, B., Fu, J., Chao, H., and Ling, H. Searching the search space of vision transformer. *Advances in Neural Information Processing Systems*, 34, 2021a.

Chen, X. and Hsieh, C.-J. Stabilizing differentiable architecture search via perturbation-based regularization. In *International Conference on Machine Learning*, pp. 1554–1565. PMLR, 2020.

Chen, X., Wang, R., Cheng, M., Tang, X., and Hsieh, C.-J. Drnas: Dirichlet neural architecture search. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=9FWas6YbmB3.

Chu, X., Zhou, T., Zhang, B., and Li, J. Fair darts: Eliminating unfair advantages in differentiable architecture search. In *European Conference on Computer Vision*, pp. 465–480. Springer, 2020.

Chu, X., Wang, X., Zhang, B., Lu, S., Wei, X., and Yan, J. Darts-: Robustly stepping out of performance collapse without indicators. In *International Conference on Learning Representations*, 2021.

Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. V. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 702–703, 2020.

Dong, C., Wang, G., Xu, H., Peng, J., Ren, X., and Liang, X. Efficientbert: Progressively searching multilayer perceptron via warm-up knowledge distillation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 1424–1437, 2021.

Dong, X. and Yang, Y. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1761–1770, 2019.

Dong, X. and Yang, Y. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations*, 2020.

Duan, Y., Chen, X., Xu, H., Chen, Z., Liang, X., Zhang, T., and Li, Z. Transnas-bench-101: Improving transferability and generalizability of cross-task neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5251–5260, 2021.

Garg, A., Saha, A. K., and Dutta, D. Revisiting neural architecture search. *arXiv preprint arXiv:2010.05719*, 2020.

Han, D., Yoo, Y., Kim, B., and Heo, B. Learning features with parameter-free layers. In *International Conference on Learning Representations*, 2022.

He, C., Ye, H., Shen, L., and Zhang, T. Milenas: Efficient neural architecture search via mixed-level reformulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11993–12002, 2020.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

He, X., Zhao, K., and Chu, X. Automl: A survey of the state-of-the-art. *Knowledge-Based Systems*, 212:106622, 2021.

Hooker, S., Courville, A., Clark, G., Dauphin, Y., and Frome, A. What do compressed deep neural networks forget? *arXiv preprint arXiv:1911.05248*, 2019.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Huang, S.-Y. and Chu, W.-T. Searching by generating: Flexible and efficient one-shot nas with architecture generator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 983–992, 2021.

Jiang, Y., Hu, C., Xiao, T., Zhang, C., and Zhu, J. Improved differentiable architecture search for language modeling and named entity recognition. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3585–3590, 2019.

Knyazev, B., Drozdzal, M., Taylor, G. W., and Romero Soriano, A. Parameter prediction for unseen deep architectures. *Advances in Neural Information Processing Systems*, 34:29433–29448, 2021.

Lee, H., Hyung, E., and Hwang, S. J. Rapid neural architecture search by learning to generate graphs from datasets. In *International Conference on Learning Representations*, 2021.

Li, L. and Talwalkar, A. Random search and reproducibility for neural architecture search. In *Uncertainty in Artificial Intelligence*, pp. 367–377. PMLR, 2020.

Li, L., Jamieson, K., Rostamizadeh, A., Gonina, E., Hardt, M., Recht, B., and Talwalkar, A. Massively parallel hyperparameter tuning. *arXiv preprint arXiv:1810.05934*, 5, 2018.

Li, L., Jamieson, K., Rostamizadeh, A., Gonina, E., Ben-Tzur, J., Hardt, M., Recht, B., and Talwalkar, A. A system for massively parallel hyperparameter tuning. *Proceedings of Machine Learning and Systems*, 2:230–246, 2020.

Li, L., Khodak, M., Balcan, M.-F., and Talwalkar, A. Geometry-aware gradient algorithms for neural architecture search. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=MuSYkd1hxRP.

Lindauer, M. and Hutter, F. Best practices for scientific research on neural architecture search. *Journal of Machine Learning Research*, 21(243):1–18, 2020.

Liu, H., Simonyan, K., and Yang, Y. Darts: Differentiable architecture search. In *International Conference on Learning Representations*, 2019.

Luo, X., Liu, D., Kong, H., Huai, S., Chen, H., and Liu, W. Surgenas: A comprehensive surgery on hardware-aware differentiable neural architecture search. *IEEE Transactions on Computers*, 2022.

Mehta, Y., White, C., Zela, A., Krishnakumar, A., Zabergja, G., Moradian, S., Safari, M., Yu, K., and Hutter, F. Nas-bench-suite: Nas evaluation is (now) surprisingly easy. In *International Conference on Learning Representations*, 2022.

Ning, X., Tang, C., Li, W., Zhou, Z., Liang, S., Yang, H., and Wang, Y. Evaluating efficient performance estimators of neural architectures. *Advances in Neural Information Processing Systems*, 34:12265–12277, 2021.

Park, D. S., Lee, J., Peng, D., Cao, Y., and Sohl-Dickstein, J. Towards nngp-guided neural architecture search. *arXiv preprint arXiv:2011.06006*, 2020.

Peng, H., Du, H., Yu, H., LI, Q., Liao, J., and Fu, J. Cream of the crop: Distilling prioritized paths for one-shot neural architecture search. *Advances in Neural Information Processing Systems*, 33, 2020.

Pourchot, A., Ducarouge, A., and Sigaud, O. To share or not to share: A comprehensive appraisal of weight-sharing. *arXiv preprint arXiv:2002.04289*, 2020.

Radosavovic, I., Johnson, J., Xie, S., Lo, W.-Y., and Dollár, P. On network design spaces for visual recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1882–1890, 2019.

Real, E., Liang, C., So, D., and Le, Q. Automl-zero: Evolving machine learning algorithms from scratch. In *International Conference on Machine Learning*, pp. 8007–8019. PMLR, 2020.

Ren, P., Xiao, Y., Chang, X., Huang, P.-Y., Li, Z., Chen, X., and Wang, X. A comprehensive survey of neural architecture search: Challenges and solutions. *ACM Computing Surveys (CSUR)*, 54(4):1–34, 2021.

Shen, J., Khodak, M., and Talwalkar, A. Efficient architecture search for diverse tasks. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=TEmAR013vK.

Shu, Y., Wang, W., and Cai, S. Understanding architectures learnt by cell-based neural architecture search. In *International Conference on Learning Representations*, 2020.

Siems, J., Zimmer, L., Zela, A., Lukasik, J., Keuper, M., and Hutter, F. Nas-bench-301 and the case for surrogate benchmarks for neural architecture search. *arXiv preprint arXiv:2008.09777*, 2020.

Stamoulis, D., Ding, R., Wang, D., Lymberopoulos, D., Priyantha, B., Liu, J., and Marculescu, D. Single-path nas: Designing hardware-e# 14; cient convnets in less than 4 hours. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2019.

Tay, Y., Dehghani, M., Abnar, S., Chung, H. W., Fedus, W., Rao, J., Narang, S., Tran, V. Q., Yogatama, D., and Metzler, D. Scaling laws vs model architectures: How does inductive bias influence scaling? *arXiv preprint arXiv:2207.10551*, 2022.

Trockman, A. and Kolter, J. Z. Patches are all you need? *arXiv preprint arXiv:2201.09792*, 2022.

Tu, R., Khodak, M., Roberts, N., and Talwalkar, A. Nas-bench-360: Benchmarking diverse tasks for neural architecture search. *arXiv preprint arXiv:2110.05668*, 2021.

Wang, D., Li, M., Gong, C., and Chandra, V. Attentive-nas: Improving neural architecture search via attentive sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6418–6427, 2021a.

Wang, L., Zhao, Y., Jinnai, Y., Tian, Y., and Fonseca, R. Neural architecture search using deep neural networks and monte carlo tree search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 9983–9991, 2020.

Wang, R., Cheng, M., Chen, X., Tang, X., and Hsieh, C.-J. Rethinking architecture selection in differentiable nas. In *International Conference on Learning Representations*, 2021b.

Wang, X., Xue, C., Yan, J., Yang, X., Hu, Y., and Sun, K. Mergenas: Merge operations into one for differentiable architecture search. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pp. 3065–3072, 2021c.

Welch, B. L. The generalization of 'student's'problem when several different population varlances are involved. *Biometrika*, 34(1-2):28–35, 1947.

Wortsman, M., Farhadi, A., and Rastegari, M. Discovering neural wirings. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pp. 2684–2694, 2019.

Wu, B., Li, C., Zhang, H., Dai, X., Zhang, P., Yu, M., Wang, J., Lin, Y., and Vajda, P. Fbnetv5: Neural architecture search for multiple tasks in one run. *arXiv preprint arXiv:2111.10007*, 2021a.

Wu, Y., Liu, A., Huang, Z., Zhang, S., and Van Gool, L. Neural architecture search as sparse supernet. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 10379–10387, 2021b.

Xie, S., Kirillov, A., Girshick, R., and He, K. Exploring randomly wired neural networks for image recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1284–1293, 2019.

Xu, Y., Xie, L., Zhang, X., Chen, X., Qi, G.-J., Tian, Q., and Xiong, H. Pc-darts: Partial channel connections for memory-efficient architecture search. In *International Conference on Learning Representations*, 2020.

Xue, C., Wang, X., Yan, J., Hu, Y., Yang, X., and Sun, K. Rethinking bi-level optimization in neural architecture search: A gibbs sampling perspective. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 10551–10559, 2021.

Yang, A., Esperança, P. M., and Carlucci, F. M. Nas evaluation is frustratingly hard. In *International Conference on Learning Representations*, 2020.

Ye, P., Li, B., Li, Y., Chen, T., Fan, J., and Ouyang, W. b-darts: Beta-decay regularization for differentiable architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10874–10883, 2022.

Ying, C., Klein, A., Christiansen, E., Real, E., Murphy, K., and Hutter, F. Nas-bench-101: Towards reproducible neural architecture search. In *International Conference on Machine Learning*, pp. 7105–7114. PMLR, 2019.

Yu, J., Jin, P., Liu, H., Bender, G., Kindermans, P.-J., Tan, M., Huang, T., Song, X., Pang, R., and Le, Q. Bignas: Scaling up neural architecture search with big single-stage models. In *European Conference on Computer Vision*, pp. 702–717. Springer, 2020a.

Yu, K., Sciuto, C., Jaggi, M., Musat, C., and Salzmann, M. Evaluating the search phase of neural architecture search. In *International Conference on Learning Representations*, 2020b.

Yu, K., Suito, C., Jaggi, M., Musat, C.-C., and Salzmann, M. Evaluating the search phase of neural architecture search. In *ICRL 2020 Eighth International Conference on Learning Representations*, number CONF, 2020c.

Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., and Yoo, Y. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6023–6032, 2019.

Zela, A., Siems, J., and Hutter, F. Nas-bench-1shot1: Benchmarking and dissecting one-shot neural architecture search. In *International Conference on Learning Representations*, 2020.

Zhang, J. and Ding, Z. Robustifying darts by eliminating information bypass leakage via explicit sparse regularization. In *Proceedings of the IEEE International Conference on Data Mining*, pp. 877–885, 2021.

Zhang, Y., Lin, Z., Jiang, J., Zhang, Q., Wang, Y., Xue, H., Zhang, C., and Yang, Y. Deeper insights into weight sharing in neural architecture search. *arXiv preprint arXiv:2001.01431*, 2020.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8697–8710, 2018.

A. Related Works

As much of the related works were already mentioned in the introduction, we highlight several fields of the closely related works separately in this section.

Topology search: A handful of previous studies focused on the search space some of which pointed out that the search space is underdeveloped compared to the rapid progress of NAS methods. Xie et al. (2019) underscored that the success of many hand-designed networks comes from the innovation of the connection pattern. They demonstrated the network topology generated by different random strategies can clearly affect performance. Shu et al. (2020) made the point that the connection pattern rather than the operation selection significantly affects the landscape of the gradient and thereby affect the convergence speed of the network. They claimed that this observation can be used as a guideline for the network design in the future. Besides, one of the most prominent difference between hyperparameter optimization and NAS is that NAS can search connection topology among different operations and layers (Zoph et al., 2018; Garg et al., 2020) which is also the key ingredient to increase the capacity of search space and find efficient networks (Real et al., 2020; Wortsman et al., 2019).

Single path (slimmable) search spaces: As another line of the gradient-based methods came up with single-path MBConv-based search spaces that are usually built on searching a combination of channel numbers, input resolutions, network depths, expansion ratio (width multipliers) where the interconnectivity patterns between operations are largely constrained (Yu et al., 2020a; Wang et al., 2021a; Dong et al., 2021; Stamoulis et al., 2019; Peng et al., 2020; Huang & Chu, 2021; Wu et al., 2021a; Shen et al., 2022). Single-path NAS regularly involves two-stage decoupled optimization (Ren et al., 2021) in which the training of supernet on ImageNet generally employ at least eight GPUs and costs hundreds of GPU hours at a minimum (Chen et al., 2021a). In contrast, DSS is mostly conducted by searching operation selections and their interconnection patterns simultaneously and costs only a few hours on a single GPU under low sample sizes. In summary, DSS is important for NAS, not only for the search of the connection topology, but also for the accessible benchmark. Still, how to design a more general, flexible and free of human bias search space will remain challenging and advantageous for the NAS community for a long time (He et al., 2021).

Benchmark on DSS: We notice some of the most recent studies of benchmark dedicated to the evaluation of multitask and transferability. These studies are often limited by the size of the search space (Duan et al., 2021), or use non-standard benchmark datasets resulting in few available baselines (Tu et al., 2021). Our benchmark is evaluated on the most commonly used standard benchmark datasets (CIFAR-10, CIFAR-100, SVHN) so that it is easy to find a large number of available baselines (handcrafted and NAS) within the latest literatures. Another irreplaceability of the DSS is that there are already extensive methods that are finetuned, provide their implementations or report their scores on DSS (He et al., 2020; Xu et al., 2020; Li et al., 2021; Chen et al., 2021b; Xue et al., 2021; Zhang & Ding, 2021; Wang et al., 2021b; Chu et al., 2021; Dong & Yang, 2019; Chen & Hsieh, 2020; Chu et al., 2020; Arber Zela et al., 2020; Lee et al., 2021), which can be used directly as the competitive baselines for DSS-based evaluations without requiring the researchers to re-implement or even re-execute the experiments. If the scores are not available out of the box, expecting researchers to reimplement multiple art baselines on a new codebase is often labor-intensive or even unachievable that also tend to get caught up in unfair comparison controversy without adequate tuning in this case. For these reasons, our study follows the configuration of the DSS benchmark to the greatest extent possible (tasks, datasets, search and evaluation protocols) so that our claim of transductive robustness is sufficiently convincing. Second and most importantly, we provide the evaluation scores of established baselines (Liu et al., 2019; He et al., 2020; Chen et al., 2021b; Li et al., 2021; Dong & Yang, 2019; Zhang & Ding, 2021; Xu et al., 2020; Chu et al., 2021; Luo et al., 2022; Ye et al., 2022) so that the researches can use the benchmark almost out of the box by only implementing their own methods and compare with the scores we provided in the main text . *Our work is not to construct a new search space and try every existing method to see which can bring us the good arch to challenge the architecture. On the contrary, our work actually embarrasses existing methods by removing their dependent artifacts, enlarging search space, searching upon different reasonable conditions*. Similar to our work, Arber Zela et al. (2020) pioneered to tailor DSS into four search spaces S1∼S4 which are specially customized to embarrass DARTS and widely used to validate the robustness of the regularization methods.

Tabular and surrogate benchmarks: NAS-BENCH-101 (Ying et al., 2019) is inappropriate as our counterpart since NB101 cannot be used by one-shot NAS methods including DARTS and its variants. NAS-BENCH-1Shot1 (Zela et al., 2020) made this issue very clear and proposed to map NB101 to three available search spaces. Nevertheless, NB1S1 was still rarely used for evaluation by current NAS methods, especially compared to its concurrent tabular work NAS-BENCH-201 (Dong & Yang, 2020), due to some internal limitations e.g. i) few available ops (only three); ii) no explicit methods ranking in the paper main text; iii) unfriendly to implement new methods. Besides, NB101 (NB1S1) is a tabular benchmark

Table 7. Softmax attends on edge in DSS as opposed to it attends on aggregation within **input end** of nodes in LHD i.e. the origin of **input**-softmax.

Search space	Inputs of s/e cell S^t	Outputs of S^t and U^t	Node x_j aggregation	Edge $g_{i,j}(x_i)$ operation
DSS	F_U^{t-1}, F_U^{t-2}	$F_S^t = \left\{x_j^t \vert j \in (n_i..N]\right\}$ where $n_i = 2$, $F_U^t = M_\oplus^{1\times1}(F_S^t)$	$x_j = \sum_{i<j} g_{i,j}(x_i)$	$g_{i,j}(x_i) = \langle \mathbf{a}_{i,j}, O_{i,j}(x_i)\rangle$ where $\mathbf{a}_{i,j} = \mathrm{softmax}(A_{i,j})$
LHD	$F_+^{t-1}, F_\oplus^{t-1}$	$F_+^t = \left\langle \mathbf{p}_+^t, \mathbf{x}^t \right\rangle$ where $\mathbf{p}_+^t = \mathrm{softmax}(P_+^t)$, $P_+^t = \left\{[\beta_+]_j^t \vert j \in (n_i..N]\right\}$ $F_\oplus^t = M_\oplus^{1\times1}\left(\mathbf{p}_\oplus^t \circ \mathbf{x}^t\right)$ where $\mathbf{p}_\oplus^t = \left\{\mathrm{sigmoid}([\beta_\oplus]_j^t) \vert j \in (n_i..N]\right\}$, $\mathbf{x}^t = \left\{x_j^t \vert j \in (n_i..N]\right\}$ and $n_i = 2$	$x_j = \langle \mathbf{a}_j, g_j\rangle$ where $\mathbf{a}_j = \mathrm{softmax}(A_j)$, $A_j = \left\{\alpha_{i,j}^m \vert i \in [1..j), m \in [1..M]\right\}$, $g_j = \{g_{i,j}(x_i) \vert i \in [1..j)\}$	$g_{i,j}(x_i) = O_{i,j}(x_i)$ $= \{o^m(x_i) \vert m \in [1..M]\}$

Table 8. Hyperparameter settings of baselines in search. DARTS settings are closely follow the released code on DSS. Specific settings of other baselines follow their released code on DSS as well. All settings keep consistent across C10, C100, SVHN.

method	batch_size	learning_rate	learning_rate_min	momentum	weight_decay	epochs	init_channels	layers	arch_learning_rate	arch_weight_decay	additional
DARTS	176	0.025	3e-4	0.9	3e-4	50	16	8	3e-4	0	-
MiLeNAS	-	-	-	-	-	-	-	-	-	-	$\lambda = 1$
DrNAS	-	-	-	-	-	-	-	-	-	-	-
GAEA-B	-	-	-	-	-	-	-	-	0.1	-	-
GAEA-E	-	-	-	-	-	-	-	-	0.1	-	-
GDAS	256	0.05	-	-	-	-	-	-	-	-	tau_min=0.1, tau_max=10
SP-DARTS	-	-	0.025	-	-	-	-	-	-	-	warmup=5, temp=0.0015
PC-DARTS	576	0.1	-	-	-	-	-	-	6e-4		warmup=15, k=4
DARTS-	-	-	-	-	-	-	-	-	-		β=1 decay scheme :linear (β →0)
β-DARTS	-	-	-	-	-	-	-	-	-		weight scheme =0→100
SurgeNAS	-	-	-	-	-	-	-	-	-		β=1 decay scheme :linear (β →0)

Table 9. Hyperparameter settings of evaluation. All baselines strictly follow the same.

batch_size	learning_rate	learning_rate_min	momentum	weight_decay	epochs	init_channels	stacked cells	data augmentations (cutout, flip, crop)
64	0.025	3e-4	0.9	3e-4	600	36 for Base, 3ops and 4out	25 for 1M, 20 for others	true for C10 and C100, false for SVHN

and the evaluation on tabular benchmarks is never enough for NAS methods due to the highly limited search space (10^5 compared to 10^{58} of DSS and 10^{84} of our LHD). Furthermore, tabular benchmarks hardly provide effective insight for the practical network design. In contrast, we have seen that more realistic DSS has inspired several studies (Shu et al., 2020; Knyazev et al., 2021). NAS-BENCH-301 (Siems et al., 2020) adopts a surrogate-based methodology on DSS that predicts performances with the performances of about 60k anchor architectures.

B. Formulations of LHD

We formalize LHD in Tabel 7 where β is the path parameter and $\circ$ denotes Hadamard product. F represents the cell outputs and subscripts U and S refers to s/e and u/e. Subscripts $+$ and $\oplus$ represent summation and concatenation. We omit the cell index t within node and edge columns and don't distinguish coessential set and vector for brevity.

C. Complexity of the Continuous DAGs

In LHD, cell accommodates five intermediate nodes with 2+3+4+5+6=20 inter-connection compound edges each of which factors in 7 operations thus a total of 2^7 combinations. For a single CSB cell output, the substructure complexity of the

Table 10. Evaluation results on CIFAR-100 and SVHN on LHD.

C100		Base			1M			3ops			4out	
Method	*val_acc* (%)	*#param*	top-1/top3	*val_acc* (%)	*#param*	top-1/top3	*val_acc* (%)	*#param*	top-1/top3	*val_acc* (%)	*#param*	top-1/top3
DARTS	69.18±3.85	0.62	72.68/71.97	70.38±4.37	1.55	74.17/73.54	70.27±2.51	0.71	73.18/72.01	70.82±2.20	0.9	72.87/72.41
DrNAS	53.92±14.67	0.5	66.90/64.16	49.65±18.55	1.54	68.89/62.56	61.69±3.59	0.53	63.77/63.38	47.21±6.21	0.85	54.69/51.35
GAEA-B	71.51±3.64	0.63	75.28/73.47	73.59±2.93	1.53	76.71/75.14	73.58±1.67	0.69	75.57/74.43	74.14±1.52	1	76.23/75.06
GAEA-E	67.81±1.03	0.86	69.12/68.47	67.06±1.95	1.54	69.90/68.26	71.07±0.40	1.06	71.51/71.35	68.10±1.13	1.21	69.34/68.86
GDAS	62.99±1.60	0.51	64.26/64.15	65.97±0.91	1.53	66.85/66.63	61.89±4.31	0.58	66.45/64.36	65.07±2.04	0.98	67.31/66.44
MiLeNAS	68.98±0.97	0.65	70.00/69.61	70.32±0.98	1.57	71.25/70.88	69.17±1.79	0.74	70.58/70.35	69.77±0.47	0.91	70.48/70.02
PC-DARTS	71.03±5.32	0.77	74.04/73.73	71.52±5.64	1.54	74.86/74.75	71.78±5.18	0.9	74.95/74.38	72.12±4.63	1.05	75.65/74.91
Random	69.14±2.79	0.7	72.08/70.93	69.34±5.78	1.54	74.46/73.13	70.66±2.24	0.79	73.18/72.17	70.71±4.17	0.97	74.84/73.64
SP-DARTS	68.99±1.13	0.52	70.69/69.60	70.97±0.71	1.55	71.89/71.44	66.88±0.92	0.55	68.08/67.47	69.15±1.60	0.84	70.90/70.06
DARTS-	68.96±2.55	0.65	71.44/70.75	67.82±3.67	1.54	72.53/70.20	61.26±6.03	0.74	67.82/64.58	57.67±6.85	0.9	65.41/62.11
β–DARTS	70.03±4.28	0.62	73.60/72.25	72.35±3.68	1.54	75.52/74.45	71.35±2.28	0.68	73.44/72.97	72.33±2.31	0.93	74.90/73.96
SurgeNAS	69.42±3.59	0.88	73.00/71.26	69.88±3.94	1.56	74.04/72.01	69.74±1.78	1.01	71.02/70.83	69.78±2.52	1.11	72.91/71.20

SVHN		Base			1M			3ops			4out	
Method	*val_acc* (%)	*#param*	top-1/top3	*val_acc* (%)	*#param*	top-1/top3	*val_acc* (%)	*#param*	top-1/top3	*val_acc* (%)	*#param*	top-1/top3
DARTS	95.77±0.70	0.58	96.36/96.26	95.94±0.59	1.55	96.52/96.35	96.00±0.50	0.68	96.43/96.34	95.58±0.58	0.87	96.20/95.96
DrNAS	96.58±0.26	0.67	96.81/96.77	96.65±0.18	1.55	96.83/96.78	96.66±0.12	0.81	96.78/96.75	96.61±0.10	1.09	96.71/96.68
GAEA-B	96.85±0.10	0.54	96.98/96.90	97.01±0.10	1.57	97.17/97.07	96.78±0.06	0.68	96.84/96.83	97.06±0.08	0.98	97.19/97.11
GAEA-E	96.68±0.07	0.87	96.76/96.73	96.63±0.02	1.54	96.66/96.64	96.58±0.07	1.06	96.67/96.64	96.63±0.06	1.05	96.68/96.67
GDAS	96.81±0.17	0.8	97.07/96.92	96.95±0.10	1.54	97.08/97.03	96.67±0.46	0.91	97.10/96.92	96.62±0.48	1.02	97.21/96.95
MiLeNAS	96.71±0.20	0.51	96.92/96.83	96.89±0.18	1.57	97.03/97.00	96.67±0.22	0.67	96.85/96.83	96.76±0.19	1	97.00/96.88
PC-DARTS	96.69±0.08	0.82	96.82/96.73	96.87±0.13	1.53	97.03/96.94	96.68±0.17	0.97	96.91/96.80	96.82±0.07	1.08	96.88/96.86
Random	96.81±0.24	0.64	97.09/96.97	96.93±0.20	1.52	97.16/97.07	96.54±0.62	0.73	96.99/96.92	96.66±0.35	0.96	97.05/96.89
SP-DARTS	96.78±0.16	0.55	97.04/96.88	97.00±0.21	1.57	97.29/97.12	96.93±0.12	0.71	97.12/96.99	96.90±0.15	0.96	97.04/97.00
DARTS-	94.45±0.59	0.54	95.17/94.86	94.66±0.47	1.56	95.15/94.96	95.23±0.24	0.63	95.59/95.37	94.70±0.46	0.85	95.17/94.96
β–DARTS	93.87±1.92	0.52	95.90/93.87	94.06±2.07	1.56	95.94/94.06	94.78±1.10	0.58	95.73/94.78	93.65±1.21	0.78	94.98/93.65
SurgeNAS	96.86±0.11	0.84	96.96/96.89	96.90±0.06	1.57	97.01/96.94	96.77±0.06	0.93	96.84/96.82	96.83±0.08	1.05	96.98/96.87

continuous space is $2^{7\times20} = 2^{140} \approx 10^{42}$ without considering graph isomorphism. Two searchable cells, normal and reduction, account for the total complexity at least $10^{42\times2} = 10^{84}$ for LHD. Softmax in DSS is applied on edge implying at least one operation will be selected. We subtract the case where no operation is selected on each edge and get the total complexity of DSS as $\left(2^7 - 1\right)^{14\times2} \approx 10^{58}$. We have to note that the post-search discretization will introduce a large amount of inductive bias and determine the valid subspace smaller than the total capacity of the continuous DAG.

D. Baseline Settings in Search and Evaluation

Hyperparameter settings in search across baselines and the consistency settings of evaluation are provided in Table 8 and Table 9 respectively. Same settings are strictly followed on all conditions.

We provide the full source code in the repository and we also report some implementation details here for self-contained. Most of the implementation of baselines are search space agnostic, so our implementations are overall closely follow the released code from their authors listed at the last column of Table 3 in the main text.

For DARTS on LHD, we closely follow the source code released on DSS. The only special clarification needed is that the parameters of the output path of both CSB and SSB are initialized sampling from $N(0, 1)$ and scaling the sample by e-3 which is in line with the initialization of the architecture parameters that representing the significance of operations in DARTS. The implementation of MiLeNAS is directly based on DARTS, except that the architecture parameters are updated by the cumulative gradients on the both training and validation sets rather than the validation set alone in DARTS. GAEA-Bilevel modifies the general gradient of DARTS to the exponential version, GAEA-ERM goes one step further and trains architecture parameters and operation weights simultaneously without splitting a validation set from training set. GDAS replaces torch.softmax with gumble-softmax in DARTS. On the other hand, DrNAS replaces torch.softmax with torch.dirichlet. We directly use the scheduler of the temperature coefficient of softmax in SP-DARTS on the LHD. We also follow the channel sampling implementation in DSS of PC-DARTS and migrate it directly to the LHD. For random sampling, we discretize the randomly sampled parameters of the operation significance and the output paths of CSB and SSB are selected randomly and independently according to the Bernoulli distribution. If the obtained network is invalid (open circuit), repeat the sampling process until a valid architecture is obtained. We use the linear decay from 1 to 0 which is the default setting in DARTS- for the shortcut of each edge within DARTS- and for the individually shortcut of each operation within SurgeNAS. The weight scheme of β-DARTS (0→100) is also consistent with the released code on DSS and NB201.

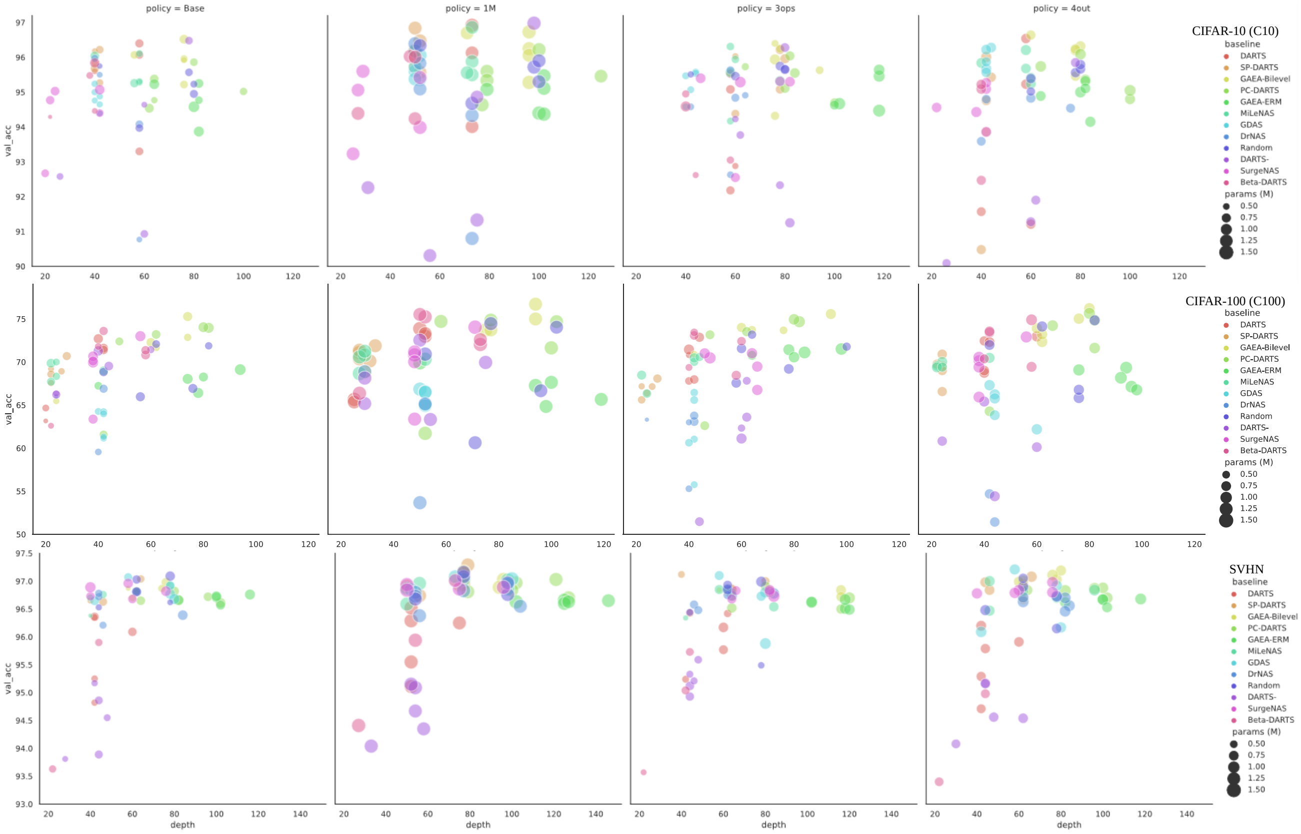


Figure 13. Distributions of the search results on *val_acc* versus depth coordinate frame. Depth refers to the number of sequential convolution layers within the longest path without counting stem. Depth has little effect on C10 performance, but deeper networks tend to achieve better C100 performance, whereas upper bound of the performance on SVHN is some kind negatively correlated with depth.

E. Additional Results of the Benchmark

We report mean and std of *val_acc*s in Table 10 as the main scores. We also report the average size of finalnets to uncover the preference of baselines in the perspective of parameter scale. We observe that some methods are trapped in rare failure cases (see DARTS and DrNAS on C10, GAEA-Bilevel and PC-DARTS on C100), so we report additional top-1 and top-3 scores in Table 10. Figure 13 illustrates the distribution of the search results in terms of depth, *val_acc* and parameter scale under the twelve conditions that the column and row of the picture group correspond to the policies and datasets respectively. The first column of Figure 14 compares baseline *val_acc*s on each policy and the second column exhibits the differences in *val_acc* between Base and other policies for each baseline to illustrate the effect of the policy on different methods.

Software version for search and evaluation of the benchmark: torch 1.9, cuda 11.1, cudnn 8.2, driver version 460.67. But we also test the search and evaluation codes and verify the empirical memory overhead on more recent version: torch 1.10, cuda 11.3, cudnn 8.3 and driver 495.44. The total number of evaluated finalnets is 540 and the footprint of both search and evaluation is about 500 GPU days.

F. Path Tuning (PT) based Result Selection

Wang et al. (2021b) proposed a new finalnet selection method based on a separated PT phase after searching to replace parameter-value-based one-off pruning. We carefully investigate the paper as well as released code[2] and have the following findings:

1. PT needs to mask and evaluate operations one-by-one in the PT phase. The idea of PT is largely NAS method agnostic

[2] https://github.com/ruocwang/darts-pt

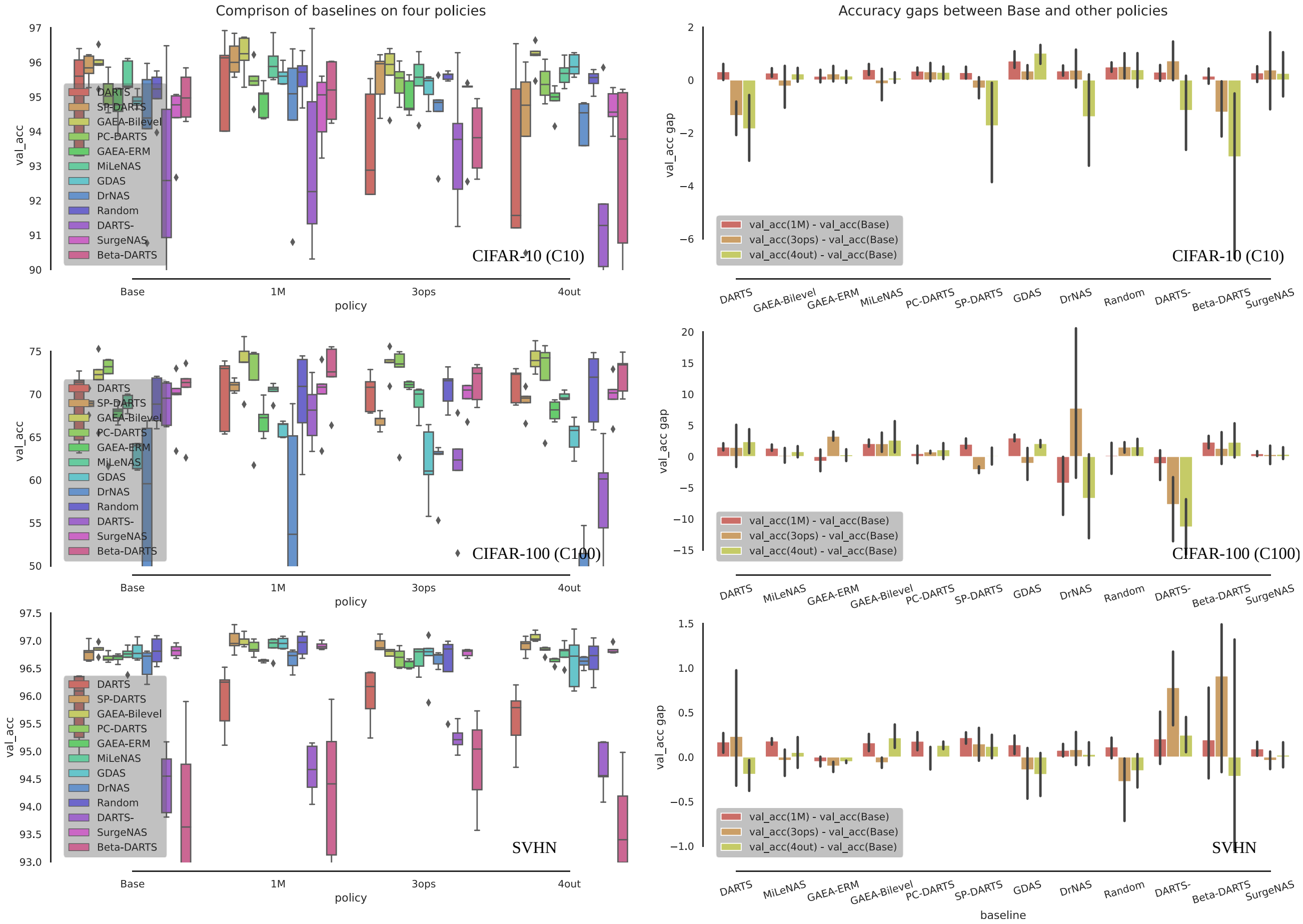


Figure 14. Baseline performances grouped by policies (left); *val_acc* gaps between Base and other policies (right).

.

but highly specific to the search space and entangled to the space design;

2. Owe to "method agnostic", we recognize that the PT can be apply to all the baselines in our benchmark but inevitably incurs non-trivial additional time overhead (See "4"), thus unfair to compare with parameter-value-based one-off pruning selection (See "5");

3. Due to "space entangled", it's non-trivial to determine many implementation details because of the difference between DSS and LHD. For example, if we need to tune the cell output, or just the operation selections? If the output path needs to be tuned, whether the tuning is done jointly with the operation selections, or separately?

4. PT needs to mask operations on each edge in forward pass to calculate the *val_acc* loss, so its computational cost is closely related to the number of operations, nodes and edges in the search space. When the space is enlarged, the computational cost will also increase linearly;

5. Apart from "4",PT has to tune supernet and select result individually for each valid space (Base, 3ops, 4out) like random search, but one-time pruning only needs to search once without any extra-search time overhead. Therefore, the overhead of PT will even exceed the search phase when it is applied to three valid spaces respectively.

For above reasons, we eliminate PT in current benchmark and leave its verification to the future work.

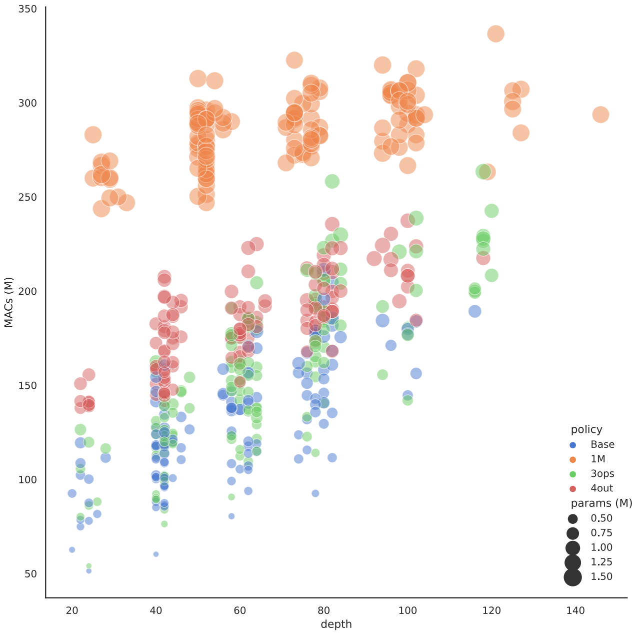


Figure 15. Distributions of the search results on MACs versus depth coordinates. The size of finalnets span a wide range of gap with a maximum 7× in terms of MACs, depth and parameters. By comparison, the depth and size are typically no more than 1.5× and 0.5× differences respectively of the search results on DSS.

G. Heuristic Regularization in Evaluation

DSS tightly couples search results and search space gives rise to that the gaps of depth and parameter scale of the finalnets released with source code never exceed 1.5× and 0.5× respectively. They elaborated a single recipe to evaluate all results on C10. In contrast, we observe a large variety of the search results in the lens of depth, flops, parameter scale due to the removable intermediate node of LHD and multiple discretization policies as shown in Figure 15 of our benchmark. This opens a new question did not appear on DSS, how to fairly evaluate search results with large differences in architecture.

Elaborating evaluation recipe for each finalnet is not our goal and quickly becomes intractable for a comprehensive evaluation. We aim to obtain reasonable and inter-comparable scores of the diverse results in the evaluation phase. Empirically, we observe that the regularization strength is the paramount factor affecting the performance of diverse architectures.

Similar to (Yang et al., 2020; Arber Zela et al., 2020) we choose to overall closely follow the evaluation recipe of DSS across different datasets similar to previous practice in addition to which we propose a (tunable and adaptive) simple protocol to adjust the intensity of regularization heuristically for various conditions. The regularization in evaluation recipe of DSS mainly involves data augmentation (Crop, flip, cutout) and drop path.

For evaluation phase of our benchmark, we adopt the same data augmentation on C10 and C100 and exclude it on SVHN. We come up a protocol $r_{\text{DP}} = ic$ to adapt the drop path rate r_{DP} under different conditions where c is the number of connections between intermediate nodes and concatenation output in the finalcell. i is a tunable parameter across datasets and discrete policies. We first set i as 0.01 for Base, 3ops and 4out on C10 and increase it by 50% for 1M due to the larger finalnet capacity. We double the value of i on C100 and SVHN due to fewer samples per class and the exclusion of data augmentation respectively which make them both more likely to be overfitted.

H. Observations of Methodological Characteristics from the Results of Benchmark

Based on the benchmark results, we can make the following observations of the methods:

1. Search results of many baselines show clustering in the depth versus *val_acc* coordinates indicating the fixed preferences of the different methods in Figure 13.

2. We can actually get rich observations for each baseline from the left column of Figure 14. For example, GAEA-ERM

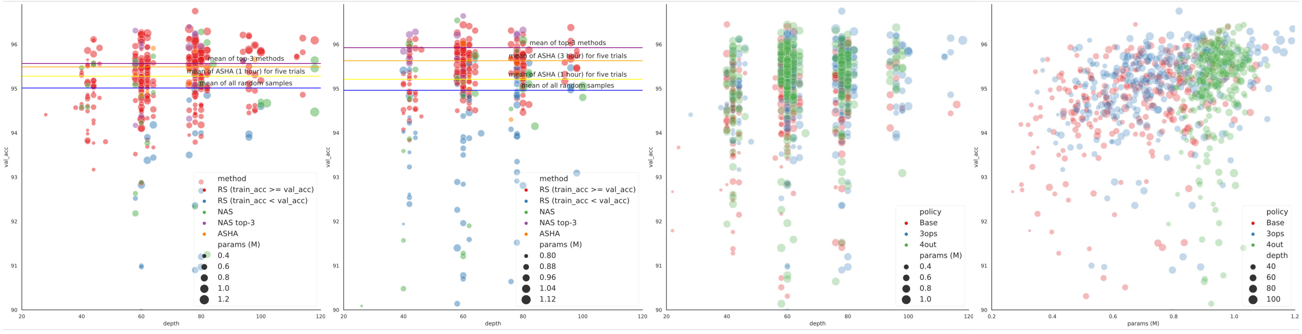


Figure 16. Random sampling and random search on C10&3ops (i) and C10&4out (ii). (iii) and (iiii) illustrate that *val_acc* is weakly correlated with depth and *#param* in all three valid spaces of LHD.

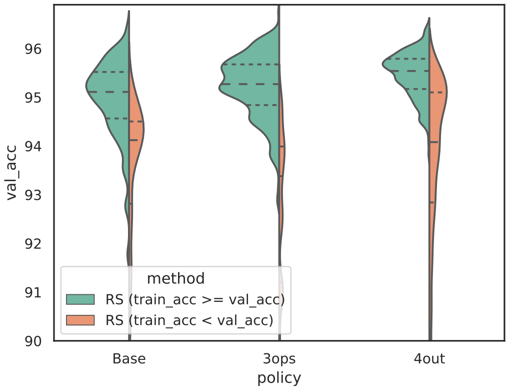


Figure 17. Base and 4out have a wider range of *val_acc* with a considerable proportion of hard-to-train samples. In comparison, 3ops can largely avoid failure cases and hard-to-train samples but the overall *val_acc* range is narrower.

shows stable performance over different seeds, MiLeNAS non-trivially reduces the *val_acc* variance on 4out, PC-DARTS is policy-insensitive on C100, DARTS is more susceptible to the initialization seeds and always have greater performance fluctuations under all conditions compared to most baselines;

3. Right column of Figure 14 shows that the baselines perform diversely on different policies. For example on C10, 4out severely deteriorates a number of baselines. GDAS, by contrast, shows remarkably superior scores on 4out than that on Base. Similarly, DrNAS and GAEA-ERM prefer 3ops but perform quite different on 4out;

4. As shown by Figure 13 and Table 10, both GAEA-ERM and PC-DARTS prefer larger and deeper cell while GDAS and SP-DARTS are just the opposite. For example on C10&Base, the average parameter scale of GAEA-ERM is 65% larger than that of GDAS, but the performance of GAEA-ERM is worse which highlights the challenging part of LHD that the methods are requisite to learn the appropriate gradient pathways autonomously rather than depending on hand-crafted skip connection;

5. SP-DARTS is one of the most performant methods on both C10 and SVHN but is poor on C100. PC-DARTS is just the opposite that performs well on C100. Failure cases are not uncommon among baselines. Both observations underpin the necessity to validate the search robustness across multiple datasets.

6. Silevel optimization is effective on stabilizing the training process shown by both GAEA-ERM and SurgeNAS. Additionally, optimizing on mixlevel can be seen as an meaningful regularization to perform consistent across different discretization policies for DARTS.

I. Random sampling and Random search

Li et al. (2018) showed ASHA to be a state-of-the-art, theoretically principled, bandit-based partial training method that

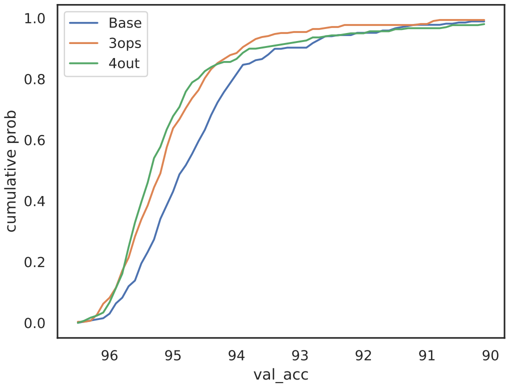


Figure 18. EDFs manifest the differences between search spaces through the curve gaps of the cumulative probability versus *val_acc* over random samples. 4out is close to 3out when the *val_acc* is $>94\%$, and close to Base when the *val_acc* is $<93\%$. We refer to (Radosavovic et al., 2019) for more information about EDFs.

outperforms leading adaptive search strategies for hyperparameter optimization. Li & Talwalkar (2020) demonstrated when implemented properly, ASHA-based random search can deliver fairly competitive baselines against NAS methods after aligning the search cost. Our experiments are based on the codebase released by Li & Talwalkar (2020)[3] where we run ASHA with a starting resource per architecture of $r = 1$ epoch and a maximum resource of 100 epochs w.r.t a promotion rate of $\eta = 4$ which indicating the top-$\frac{1}{4}$ of architectures will be promoted in each round and trained for $4\times$ more resources. We refer to Li & Talwalkar (2020) for more details of the random search.

The results of random sampling (RS) and random search (ASHA) on C10&3ops and C10&4out are provide in Figure 16(i) and (ii). We also illustrate all random samples on *val_acc* versus depth coordination in Figure 16(iii) and *val_acc* versus *#param* coordination in Figure 16(iiii) respectively. Figure 17 exhibits the proportion of sample accuracy distribution in different search spaces in which hard-to-train samples (*train_acc*<*val_acc*) are particularly identified. Radosavovic et al. (2019) proposed to characterize the distributions of architecture spaces through empirical distribution functions (EDFs) in a cumulative probability versus *val_acc* coordination as shown in Figure 18.

[3] https://github.com/liamcli/randomNAS_release