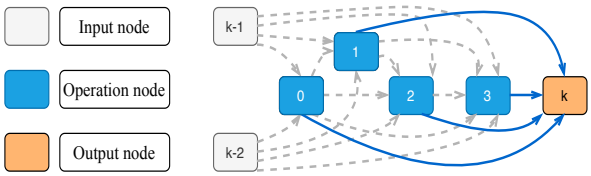


Figure 4. Description of DARTS search space.

APPENDIX:

A. Search Spaces and Experimental Setting

In our experiments, we consider two scenarios, NAS benchmark datasets and the common DARTS space, to analyze the proposed framework FreeDARTS. The high computational cost in evaluation is a major obstacle when analyzing and reproducing differentiable NAS methods. To alleviate this issue, several benchmark datasets have been recently published [16, 40, 48, 51], where the ground-truth for all candidate architectures in the benchmark datasets is known. The NAS-Bench-201 dataset [16] is a popular NAS benchmark dataset to analyze differentiable NAS methods. The search space in NAS-Bench-201 contains four nodes with five associated operations, resulting in 15,625 cell candidates, where the performance of CIFAR-100, CIFAR-100, and ImageNet for all architectures in this search space are reported. The NAS-Bench-101 [48] is another famous NAS benchmark dataset, which is much larger than NAS-Bench-201 while only the CIFAR-10 performance for all architectures are reported. More important, the architectures in NAS-Bench-101 contain different number of nodes, which makes it impossible to build a generalized supernet for one-shot nor differential NAS methods. To leverage the NAS-Bench-101 for analyzing the differentiable NAS methods, NAS-Bench-1shot1 [51] builds from the NAS-Bench-101 benchmark dataset by dividing all architectures in NAS-Bench-101 into 3 different unified cell-based search spaces, which contain 6240, 29160, and 363648 architectures, respectively. The architectures in each search space have the same number of nodes and connections, making the differentiable NAS could be directly applied to each search space. We choose the third search space in NAS-Bench-1shot1 since it is much more complicated than the remaining two search spaces.

As to the most common search space in NAS, DARTS needs to search for two types of cells: a normal cell α_{normal} and a reduction cell α_{reduce}. Cell structures are repeatedly stacked to form the final CNN structure. There are only two reduction cells in the final CNN structure, located in the 1/3 and 2/3 depths of the network. There are seven nodes in each cell: two input nodes, four operation nodes, and one output node. Each operation node selects two of the previous nodes' output as input nodes in this search space. Each input node will select one operation from $|\mathcal{O}| = 8$ candidate operations. Fig. 4 describes a unified convolutional search space in DARTS. The common practice in DARTS is to search on CIFAR-10, and the best searched cell structures are directly transferred to CIFAR-100 and ImageNet. The experimental settings on DARTS space in this paper are following the common DARTS setting. We conduct the architecture search with 5 different *random seeds*, and the best one is selected after the evaluation on CIFAR-10, which is then transferred to CIFAR-100 and ImageNet. The architecture evaluation for CIFAR-10 and CIFAR-100 are on a single GPU with batch size 96, while for ImageNet is performed on 2 GPUs. In addition, we adjust the number of filter in the evaluation to make the model sizes similar for fair comparison. We use a linear learning rate scheduler with following PDART [10] and PCDARTS [47] to use a smaller slope in the last five epochs for the architecture evaluation on ImageNet.

B. Ablation Study on the Saliency Metrics

In our BaLeNAS-TF, we utilize three train-free saliency metrics, SNIP, GraSP, and Synflow, as proxies for the architecture selection from the optimized distribution. In Table 4, we considered different number of sample size for our BaLeNAS-TF when combined with the three saliency metrics. As shown in Table 4, combined with different train-free proxies, our BaLeNAS-TF achieve higher performance than the original BaLeNAS when the sample size is 10. However, when increasing the sample size, we can see a sharp drop for BaLeNAS-TF with SNIP and GraSP, showing the two metrics are not appropriate metrics to for the architecture selection. On the contrary, the SynFlow, also adopted by our BaLeNAS-TF, shows a clear improvement with the sample size from 10 to 100, implying that this proxies is more reliable for the architecture selection.

C. Ablation Study of MCMC on NAS-Bench-201

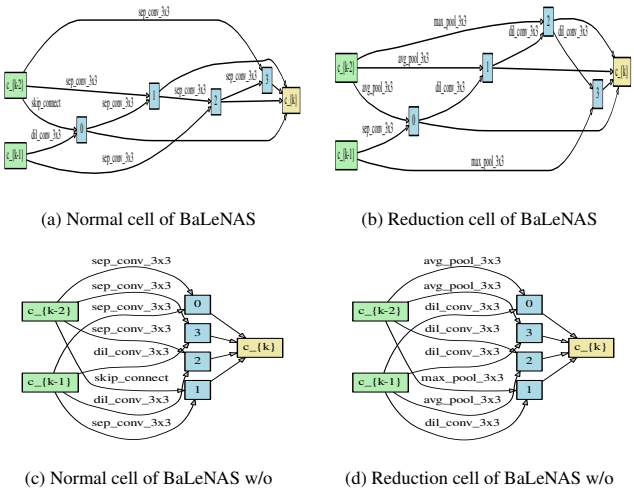
As we described in Section 3.2, one key additional hyperparameter in BaLeNAS is the sampling number M in MCMC, and this subsection investigates how this hyperparameter affects the performance of BaLeNAS. Table 5 summarizes the performance our BaLeNAS (2nd) with different number of MCMC sampling. As shown, our BaLeNAS is very robust to the number of MCMC sampling, where BaLeNAS achieves excellent results under different scenarios, outperforming most existing NAS baselines. An interesting observation is that the performance of BaLeNAS increase with multiple samplings when $M < 4$ in MCMC, and $M = 3$ achieves the best performance. Theorem 1 in [22] points out that VAdam with $M > 1$ will converge fast while might result in slightly less exploration. The exploration and exploitation can be balanced by the MCMC sample size. A detailed explanation can be found in the Section 3.4 of [22].

Table 4. Zero-cost NAS and FreeDARTS with different saliency metrics on NAS-Bench-201.

Method	Sample Size	CIFAR-10 Valid(%)	CIFAR-10 Test(%)	CIFAR-100 Valid(%)	CIFAR-100 Test(%)	ImageNet-16-120 Valid(%)	ImageNet-16-120 Test(%)
BaLeNAS	-	91.32±0.09	94.02±0.14	71.53±0.08	71.93±0.27	45.39±0.17	45.48±0.39
BaLeNAS with SNIP	10	90.95±0.39	93.85±0.22	71.37±0.35	71.48±0.52	46.04±0.47	46.03±0.41
	50	88.23±2.18	92.56±1.18	68.26±2.74	64.58±3.18	27.13±9.20	35.23±10.3
	100	86.04±0.00	91.37±0.00	65.52±0.00	67.77±0.00	36.33±0.00	24.97±0.00
BaLeNAS with Grasp	10	91.10±0.23	93.94±0.05	72.03±0.53	72.00±0.06	45.26±0.56	44.67±1.54
	50	90.56±0.76	93.72±0.16	71.52±1.03	70.62±1.43	45.01±0.81	44.92±1.64
	100	89.01±0.78	92.32±1.25	67.86±2.61	67.32±1.85	40.29±3.91	39.84±3.43
BaLeNAS with SynFlow	10	91.52±0.04	94.08±0.13	72.37±0.53	72.55±0.42	45.34±0.23	45.82±0.30
	50	91.52±0.04	94.33±0.03	72.67±0.41	72.95±0.28	46.14±0.23	46.54±0.36
	100	91.52±0.04	94.33±0.03	72.67±0.41	72.95±0.28	46.14±0.23	46.54±0.36

Table 5. Ablation study on the MCMC sampling size on NAS-Bench-201.

MCMC number	CIFAR-10 Valid(%)	CIFAR-10 Test(%)	CIFAR-100 Valid(%)	CIFAR-100 Test(%)	ImageNet-16-120 Valid(%)	ImageNet-16-120 Test(%)
$M = 1$	90.52±0.09	93.33±0.04	70.67±0.08	70.95±0.27	44.39±0.47	44.32±0.39
$M = 2$	90.71±0.12	93.75±0.87	71.25±0.92	71.43±0.45	44.63±0.55	45.05±0.95
$M = 3$	91.32±0.09	94.02±0.14	71.53±0.08	71.93±0.27	45.39±0.17	45.48±0.39
$M = 4$	90.03±0.96	93.04±1.09	68.80±1.46	69.20±1.86	43.09±2.93	43.21±2.88
Random baseline	83.20±13.28	86.61±13.46	60.70±12.55	60.83±12.58	33.34±9.39	33.13±9.66
DARTS (2nd)	37.51±3.19	53.89±0.58	13.37±2.35	13.96±2.33	15.06±1.95	14.84±2.10
optimal	91.61	94.37	73.49	73.51	46.77	47.31



(a) Normal cell of BaLeNAS (b) Reduction cell of BaLeNAS

(c) Normal cell of BaLeNAS w/o (d) Reduction cell of BaLeNAS w/o

Figure 5. Examples of searched cells by BaLeNAS and BaLeNAS without regularizatons (BaLeNAS w/o).

D. Searched Architectures Visualization

Fig.5 plots the searched architectures on DARTS space by BaLeNAS and BaLeNAS-TF. We could observe that, our BaLeNAS tends to obtain "shallow" architectures, which is also observed by several existing works [34, 54]. As we know the shallow architectures are easier to train and usually perform excellently in the small dataset, implying that the differentiable NAS methods prefer those "shallow" architectures if we only utilize the validation accuracy as the indicator. However, the performance of those "shallow" architectures on the large dataset is not as competitive as on the small dataset, indicating poor transferability. These results suggest the importance of introducing other indicator to differentiable NAS for architecture search, especially in the complicated real-world search space, to help finding more robust architectures. In contrast, as shown in Fig.5, our BaLeNAS-TF can found "deeper" architectures as it does not only rely on the validation accuracy for the architecture, but also another saliency metric. We can find a similar phenomenon in the NAS-Bench-201 search space that, even though DrNAS achieves near-optimal results on CIFAR-10, while our BaLeNAS-TF outperform it on the larger dataset.

Related Works

Unlike directly optimizing the architecture parameters, several recent works formulate the differentiable NAS as a distribution learning problem by relaxing architecture param-

eters into different distributions. SNAS [46] and GDAS [15] formulate the architecture as a discrete distribution with concrete relaxation and utilize the Gumbel-softmax trick to obtain the discrete architecture. DrNAS [9] treats the continuous architecture parameters as random variables being modeled by a learnable Dirichlet distribution. This distribution is parameterized by a concentration parameter β, which controls the sampling behavior and is optimized via pathwise derivative estimators [20]. Zheng et al. [57] consider the whole search space as a joint multinomial distribution and learn the probabilities of candidate operations among all nodes based on the multinomial distribution learning. A common point in these previous methods is that they formulate the architecture parameters as simple distributions in which only one parameter needs to be learned. In this way, these learning paradigms are easy to fit with existing DARTS codebases.

Rather than considering the above distributions, this paper considers the more general Gaussian distributions for the architecture parameters. By leveraging natural-gradient variational inference (NGVI), the architecture parameter distribution could be learned with by only updating a natural parameter λ during the search. The most relevant work to ours is BayesNAS [58], which also considers the Bayesian learning approach for neural architecture search. BayesNAS models the architecture parameters with hierarchical automatic relevance determination (HARD) priors, while which casts NAS as a model compression problem. The architecture parameters is formulated as $q(\theta) \sim \mathcal{N}(\mu, \psi^{-1})$, where ψ is a hyperparameter to tune rather than a parameter to learn. Furthermore, not only the architecture parameters are formulated as distributions, the supernet in BayesNAS is also formulated as a Bayesian Neural Network, which is hard to train and BayesNAS could only train the supernet for one epoch. Differently, our BaLeNAS only replaces the Adam optimizer with the Variational Adam optimizer for architecture optimization in the DARTS codebase, and keeps the supernet the same. In this way, our BaLeNAS is easy to be applied to most existing differentiable NAS codebases with minimal modifications.