

# Differentiable Architecture Search Meets Network Pruning at Initialization: A More Reliable, Efficient, and Flexible Framework

Miao Zhang<sup>1</sup> Wei Huang<sup>2</sup> Steven Su<sup>2</sup> Shirui Pan<sup>3</sup> Xiaojun Chang<sup>4</sup> Bin Yang<sup>1</sup> Gholamreza Haffari<sup>2</sup>  
<sup>1</sup>Aalborg University <sup>2</sup>UTS <sup>3</sup>Monash University <sup>4</sup>RMIT University

## Abstract

Although Differentiable ARchiTecture Search (DARTS) has become the mainstream paradigm in Neural Architecture Search (NAS) due to its simplicity and efficiency, more recent works found that the performance of the searched architecture barely increases with the optimization proceeding in DARTS, and the final magnitudes obtained by DARTS could hardly indicate the importance of operations. The above observation reveal that the supervision signal in DARTS may be a poor or unreliable indicator for the architecture search, inspiring an interesting and promising direction: **can we measure the operation importance without any training under the differentiable paradigm?** We provide an affirmative answer by customizing the NAS as a network pruning at initialization problem. With leveraging recently-proposed synaptic saliency criteria in the network pruning at initialization, we seek to score the importance of candidate operations in differentiable NAS without any training, and proposed a novel framework called training free differentiable architecture search (**FreeDARTS**) accordingly. We show that, without any training, FreeDARTS with different proxy metrics can outperform most NAS baselines in different search spaces. More importantly, FreeDARTS is extremely memory-efficient and computational-efficient as it abandons the training in the architecture search phase, enabling FreeDARTS to perform architecture search on a more flexible space and eliminate the depth gap between architecture search and evaluation. We hope our work inspires more attempts in solving NAS from the perspective of pruning at initialization.

## 1. Introduction

Neural Architecture Search (NAS) [9, 20, 31, 46] automates the neural network design process and has, therefore, received broad attention. The downside is that it comes with an extremely high demand for computation power, where the early NAS methods cost more than thousands of GPU days to search for a promising architecture [30, 50]. To improve the efficiency, many recent studies have been shifted to reducing the search costs [3, 17, 45], and one of the most

popular paradigms is termed as *Differentiable ARchiTecture Search* (DARTS) [25] framework. DARTS adopts the continuous relaxation to convert the operation selection problem into the continuous magnitude optimization for a set of candidate operations, which is then formulated as a bi-level optimization problem with alternatively optimizing the architecture parameters and model weights by gradient descent in a weight-sharing supernet.

Although DARTS provides a concise and efficient framework for NAS, more recent works find it is still unreliable [6, 32, 37] as DARTS usually observes a performance collapses with search progresses. A recent work by Wang *et al.* [37] showed that the magnitude of architecture parameter obtained by DARTS after supernet training is fundamentally wrong, which could hardly indicate the importance of operations. More interesting, several several works [23, 43] utilized simple early-stopping strategies to interrupt the supernet training during the search, while which could significantly improve the performance of DARTS. These empirical observations show that the supernet training seems to deteriorate the performance with search progresses. On the other hand, although most NAS alternatively optimize the architecture parameters and model weights by the supervision signal, (e.g., training and validation accuracy), the recent emerged unsupervised NAS without label information [21, 24, 41, 47] has been validated to achieve comparable performance to supervised NAS methods. Furthermore, DARTS is memory-unfriendly as it requires to train the whole supernet in each step of architecture search. Due to the memory limitation, DARTS is only able to search on a shallow network to get a cell structure, and then evaluate in a deeper network through stacking more cells. This brings an issue named the *depth gap* [8]. The above observations inspire us to discard the supernet training and abandon the supervision signal in DARTS, with raising the question:

- **Can we measure the operation importance without any training under the differentiable paradigm?**

We provide an affirmative answer by customizing the NAS as a network pruning problem, corresponding to a promising yet underexplored direction of neural architec-

ture search at initialization [1, 5, 27], which we formalize in Section 3. This work is inspired by recent research on network pruning-at-initialization [19, 35, 36], which utilizes a class of score metrics, called as *synaptic saliency* [35] or *connection sensitivity* [18], to measure the importance of weights in neural networks at initialization. Instead of using the weight magnitude after training to measure the importance for network pruning, these works argue to preserve the *synaptic flow* [35] or *information flow* [36] through the network at initialization. Accordingly, rather than calculating the magnitudes of operations after training as DARTS, we intend to utilize the saliency criteria to measure the operation importance at initialization, where we propose a novel framework called *training free differentiable architecture search* (**FreeDARTS**) with leveraging these techniques. Built on *synaptic saliency metrics* in network pruning at initialization, we adapt these metrics to score operations in differentiable NAS in a training-free manner, called *operation saliency metrics*. Integrating with the proposed metrics, FreeDARTS can perform architecture search extremely efficiently by discarding the memory and computation-consuming supernet training part, and achieve competitive or even better results against SOTA. More importantly, with the efficiency of FreeDARTS, we can perform architecture search on a more complicated space, and eliminate the depth gap between architecture search and evaluation. Rather than emphasizing the effectiveness of the proposed framework, we hope our work opens up a promising direction for differentiable NAS, resolving the architecture search from the perspective of network pruning at initialization, as which provides a more reliable, efficient, and flexible framework. A summary of our main contributions follows.

- Firstly, this paper formulates the differentiable architecture search as network pruning at initialization and builds a generalized framework called *training free differentiable architecture search* (**FreeDARTS**). We adapt several existing metrics on network pruning at initialization into the proposed FreeDARTS framework, called *operation saliency metrics*. We empirically verified the effectiveness of the proposed metrics for differentiable NAS, and the experimental results demonstrate that the proposed framework is a promising and **reliable** solution to differentiable neural architecture search, which achieves competitive performance on different benchmark datasets and DARTS search space.
- Secondly, our FreeDARTS is extremely **efficient**, which completes the architecture search in seconds. On the NAS-Bench-201, our FreeDARTS completes the architecture search with only **0.6s**, achieving 93.66, 70.78%, 46.53% test accuracy on three datasets, respectively. On the more complicated DARTS space, FreeDARTS completes the search with **1.5s**, and the

best searched architecture returns competitive results on CIFAR-10, CIFAR-100, and the ImageNet datasets, with test error 2.45%, 16.85%, and 24.4%, respectively.

- Thirdly, due to the memory and computation efficiency, we are able to apply FreeDARTS on a more **flexible** search space, where the depth of network in the architecture search and evaluation can be the same. In addition, rather than searching on a small dataset and then transferring to a large dataset for evaluation, our FreeDARTS can be directly applied on the large dataset for architecture search. The above advantages all show the **flexibility** of the proposed framework.

## 2. Preliminary

### 2.1. A Review of NAS

Neural Architecture Search (NAS) focuses on automating the process of neural network design, which has attracted increasing attention recently as it relieves human experts from the labor-intensive neural network design process. However, the early NAS methods suffer from heavy computational demand. To relieve the computational burden, more recent studies shift to reducing the search cost [2, 45], and the *weight-sharing* is one of the most well-known paradigm [13, 22, 29].

**Differentiable ARchiTecture Search (DARTS)** [25] is built on weight-sharing NAS, with further adopting the continuous relaxation to convert the operation selection into the continuous magnitude optimization, thus enable gradient descent for architecture search. The DARTS can be formulated as a bi-level optimization problem:

$$\begin{aligned} \min_{\alpha} \quad & \mathcal{L}_{val}(W^*(\alpha), \alpha) \\ \text{s.t.} \quad & W^*(\alpha) = \operatorname{argmin}_W \mathcal{L}_{train}(W, \alpha), \end{aligned} \tag{1}$$

where  $\alpha$  is the continuous architecture parameter representation and  $W$  is the supernet weights.

**Failure of DARTS.** Despite notable benefits on computational efficiency from DARTS, more recent works find it is still unreliable [6, 43] that directly optimizing the architecture magnitudes. For example, a recent work by Wang *et al.* [37] showed that the magnitude of architecture parameter obtained by DARTS after supernet training is fundamentally wrong, where the optimized magnitude could hardly indicate the importance of operations. Rather than utilizing the optimized magnitudes by DARTS to indicate the operation strength, the authors find the perturbation-based operation influence on supernet can more consistently extract significance of operation than magnitude-based counterparts. In addition, DARTS is unable to stably obtain excellent solutions as which yields deteriorative architectures with the search proceeding, performing even worse than random search in some cases [32]. Zela *et al.* [43] interrupt the search based

on the dominant eigenvalue of the Hessian matrix, and Liang *et al.* [23] introduce another simple “early stopping” criteria, where the search procedure ends as one cell has two or more *skip-connection* operations. They both show the supervised learning paradigm maybe harmful for finding a promising architecture, especially with the search proceeding.

**Unsupervised NAS** conducts architecture search with the absence of labels. Liu *et al.* [24] utilize unsupervised objectives to estimate the performance of architectures during the architecture search, where they found that the architectures searched by unsupervised objectives are comparable in performance to those architectures searched by supervised objectives. To avoid using the human-annotated labels, RLNAS [47] randomly generates labels for the architecture search, where a customized angle metric [15] is introduced to measure the distance between trained and initialized weights, to estimate the convergence speed of the corresponding architecture. On the other hand, Yan *et al.* [41] utilize unsupervised pretraining for a better architecture representation to assist the subsequent supervised search. The above works all show that the supervised signal maybe not the key to find the promising architectures.

**Train free NAS** tries to identify promising architectures at initialization without incurring training. Mellor *et al.* [27] empirically find that the correlation between sample-wise input-output Jacobian can indicate the architecture’s test performance, and propose using the Jacobian to score a set of randomly sampled models with randomly initialized weights, which greedily chooses the model with the highest score. TE-NAS [5] utilizes the spectrum of NTKs and the number of linear regions to analyzing the trainability and expressivity of architectures, and leverage the perturbation-based architecture selection as [37] for the supernet prune. Since the calculation of NTK is not easy, TE-NAS can only improve efficiency to a limited extent. Zero-cost NAS [1] partially shares the same motivation as us, which both utilize saliency metrics in the network pruning at initialization for architecture search. A concurrent work zero-cost-pt NAS [38] further extends the zero-cost NAS for differentiable NAS, through leveraging perturbation as [5, 37] for the supernet prune. Different from zero-cost NAS or zero-cost-pt NAS that measures the whole network through summing saliency scores of all parameters  $\theta$  in the architecture, our FreeDARTS is designed to directly measure the importance of candidate operations without perturbation-based selection.

## 2.2. Connecting NAS with Network Pruning

Network pruning is an effective way to compress overparametered neural networks by removing parameters with minimal performance degradation. The final discretization phase of DARTS, selecting a discrete architecture from an overparametered supernet based on the magnitudes of op-

erations, can be considered as an operation-level network pruning [37]. The network pruning can happen after training, during the training, and before the training (a.k.a, at initialization) [35]. In this subsection, we will review existing NAS literature that conduct pruning in different phases.

**Pruning after training.** Most conventional network pruning methods operate on a trained network to identify the redundant weights based on different criteria, where those redundant weights are removed with least degrading the performance. Magnitude based pruning [14,16] is the most common paradigm, which directly measures the importance of each weight based on its value. The well-known DARTS [25] also adopts this pruning paradigm in the final discretization phase, which measures the magnitude of candidate operations after supernet training to prune the supernet at once. However, more recent work [37] found that the magnitude of architecture parameters does not necessarily indicate how much the operation contributes to the supernet’s performance, and the pruning many weak connections at once can hardly find the promising architecture. Alternatively, Wang *et al.* [37] proposed a perturbation-based architecture selection to measure each operation’s influence on the supernet after training, and only prune one edge at once before fine-tuning the supernet.

**Pruning during training.** Rather than pruning all redundant weights at once after training, some works try to incorporate pruning into the training procedure [26,34]. Based on this motivation, ASAP [28] proposes a differentiable annealable search space, enabling gradually pruning inferior operations during the search. Since it reduced the number of candidate operations during the search, ASAP could also accelerate the search. Similarly, PDARTS [8] also prunes the cell candidate operations during the search, while it progressively increases the network depth, to alleviate the *depth gap* during the architecture search and evaluation.

**Pruning at initialization.** Recent works on network pruning shows that the randomly initialized neural network can be pruned without incurring any training [18], where SNIP [19], GraSP [36], and SynFlow [35] are three well-known metrics for the network pruning at initialization. These scores are designed to measure the importance of weights  $\theta$  in a networks. Zero-cost NAS [1] extend the above three saliency metrics as a series zero-cost proxies to assist existing NAS algorithms, which utilizes the zero-cost proxies to *warmup* different search algorithms, e.g., initializing population or controller for aging evolution NAS and RL based NAS, respectively. Similar to [1], our FreeDARTS also leverages these saliency metrics for NAS, while one strength of FreeDARTS over [1] is that FreeDARTS could obtain a valid architecture along, instead of assisting NAS methods.

### 3. Methodology

Rather than alternatively training supernet weights and architecture parameters before pruning the supernet based on the operation magnitude as DARTS, this paper tries to investigate searching a promising architecture at initialization. More specifically, our goal is to explore to score the importance of operations for the supernet pruning, without any training nor even labels, thus significantly improve the efficiency. In this section, we first introduce a class synaptic saliency based score metrics [1, 19, 35, 36], which are used in the network pruning at initialization. Instead of utilizing the saliency metrics to score the supernet weights  $W(\theta)$  as zero-cost NAS [1], we focus on the importance of  $\alpha$  for the architecture pruning, where we then accordingly design a class of saliency metrics to score the operations in the supernet. Those saliency metrics are training-free, which enables us to prune the supernet without training.

#### 3.1. Revisit Synaptic Saliency Metrics

The synaptic saliency criteria [35] are designed to measure the change of loss functions when a specific parameter  $\theta$  is removed from the neural network, to imply the importance of parameters. The synaptic saliency based score metrics are generally described as the form of Hadamard product:

$$S(\theta) = \frac{\partial \mathcal{R}}{\partial \theta} \odot \theta, \quad (2)$$

where  $\mathcal{R}$  is a class of loss functions. SNIP [19] is the first work based on this saliency criteria to perform network pruning at initialization. Utilizing the common training loss  $\mathcal{L}$ , SNIP defines a notion of connection sensitivity to measure how removing each parameter affects the loss at initialization:

$$\mathcal{S}_{snip}(\theta) = \left| \frac{\partial \mathcal{L}}{\partial \theta} \odot \theta \right|. \quad (3)$$

Rather than minimizing the effects on the loss function after pruning, GraSP [36] proposes to preserve the gradient flow, a.k.a. the change in the gradient norm:

$$\mathcal{S}_{grasp}(-\theta) = -(H \frac{\partial \mathcal{L}}{\partial \theta}) \odot \theta, \quad (4)$$

where  $H$  is the Hessian matrix. As described, SNIP and GraSP are data-dependent as they need to calculate the training loss based on the labels. Different from them, SynFlow [35] proposes to identify important weights without any labels at initialization. Instead of using the normal label-dependent training loss function  $\mathcal{L}$ , SynFlow introduces a new loss function with setting the input as the all-ones vector  $\mathbf{1}$ , and the loss function is described as:

$$\mathcal{R}_{SF} = \mathbf{1}^T \left( \prod_{l=1}^L |\theta^{[l]}| \right) \mathbf{1}, \quad (5)$$

where SynFlow utilizes the element-wise absolute  $|\theta^{[l]}|$  to make the corresponding synaptic saliency score positive to avoid the layer-collapse [35]:

$$\mathcal{S}_{SF}(\theta) = \frac{\partial \mathcal{R}_{SF}}{\partial \theta} \odot \theta. \quad (6)$$

Zero-cost NAS [1] extends the above saliency metrics as zero-cost proxies to score an architecture, through summing scores of all parameters  $\theta$  in the architecture. Since these scores can be obtained without any training, zero-cost proxies [1] could assist NAS by *warmup* different search algorithms, e.g., initializing population or controller for aging evolution NAS and RL based NAS, respectively. However, we could not follow the paradigm of zero-cost NAS to measure the importance of operations, since there are several candidate parameter-free operations without affiliated parameters, e.g., *zero*, *skip-connection*, *pooling*.

#### 3.2. Adapt Saliency Metrics for Differentiable NAS

Different from zero-cost NAS [1] assisting existing NAS algorithms, we try to design a saliency metric to prune the supernet and obtain the promising architecture at initialization without any training. Rather than score the importance of weights, we should design a metric to score the importance of operations  $\alpha$ . As described in Eq.(1), there are two parts of parameters needed to be optimized in DARTS, supernet weights  $W(\theta)$  and architecture parameters  $\alpha$ . The final stage of DARTS can be seen as an operation level pruning [37], where DARTS calculates the magnitude of operations after training to prune the supernet. The intuitive purpose of this paper is to measure the importance of each operation, where we try to utilize the training-free saliency scores to directly measure the importance of  $\alpha$  at initialization in our FreeDARTS, which we call as *operation saliency metrics*. Accordingly, the SNIP based operation saliency metric to score the importance of  $\alpha$  in our FreeDARTS is defined as:

$$\mathcal{F}_{snip}(\alpha) = \left| \frac{\partial \mathcal{L}}{\partial \alpha} \odot \alpha \right|. \quad (7)$$

where  $\mathcal{L}$  is the  $\mathcal{L}_{val}$  in Eq.1 while based on an initialized supernet  $W$ . Similarly, GraSP based operation saliency metric is defined as:

$$\mathcal{F}_{grasp}(-\alpha) = -(H \frac{\partial \mathcal{L}}{\partial \alpha}) \odot \alpha. \quad (8)$$

To further make our method label-agnostic, we also leverage SynFlow to design a label-agnostic loss function:

$$\mathcal{R}_{SF}(\alpha) = \mathbf{1}^T \left( \prod_{l=1}^L |W(\theta, \alpha)^{[l]}| \right) \mathbf{1}, \quad (9)$$

where we need apply the element-wise absolute on weights  $W(\theta, \alpha)$  as SynFlw to make  $\mathcal{F}_{SF}$  positive. The corresponding SynFlow based operation saliency metric for  $\alpha$  is then



Table 1. Comparison results with NAS baselines on NAS-Bench-201.

Method	CIFAR-10		CIFAR-100		ImageNet-16-120		Search Cost (GPU sec.)
	Valid(%)	Test(%)	Valid(%)	Test(%)	Valid(%)	Test(%)	
SETN [10]	84.04±0.28	87.64±0.00	58.86±0.06	59.05±0.24	33.06±0.02	32.52±0.21	31010
GDAS [11]	89.88±0.33	93.40±0.49	70.95±0.78	70.33±0.87	41.28±0.46	41.47±0.21	28925.91
DARTS (1st) [25]	39.77±0.00	54.30±0.00	15.03±0.00	15.61±0.00	16.43±0.00	16.32±0.00	10889.87
DARTS (2nd) [25]	39.77±0.00	54.30±0.00	15.03±0.00	15.61±0.00	16.43±0.00	16.32±0.00	29901.67
PC-DARTS [40]	89.96±0.15	93.41±0.30	67.12±0.39	67.48±0.89	40.83±0.08	41.31±0.22	10023.
SNAS [39]	90.10±1.04	92.77±0.83	69.69±2.39	69.34±1.98	42.84±1.79	43.16±2.64	32345
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	-
NASWOT [27]	89.16±1.13	92.45±1.12	68.53±2.01	68.66±2.02	41.13±3.94	41.35±4.08	30.01
Zero-Cost NAS [1]	90.19±0.66	93.45±0.28	70.55±1.61	70.73±1.36	43.24±2.52	43.64±2.42	115.2
Zero-Cost-PT NAS [38]	-	93.75±0.00	-	71.11±0.00	-	41.43±0.00	647.5
TE-NAS* [5]	89.99±0.40	93.28±0.25	69.19±0.63	69.62±0.71	43.72±2.06	44.29±1.97	1558
<b>FreeDARTS</b>	90.39±0.30	93.52±0.11	70.96±0.51	70.70±0.08	44.68±1.67	45.28±1.73	<b>0.6</b>
<b>optimal</b>	91.61	94.37	74.49	73.51	46.77	47.31	-

“\*\*” indicates the results reproduced with the same *seeds*. We use the Synflow metric in this experiment. Our best single run achieves **93.66%**, **70.78%**, and **46.53%** test accuracy on three datasets, respectively.

defined as:

$$\mathcal{F}_{\text{SF}}(\alpha) = \frac{\partial \mathcal{R}_{\text{SF}}}{\partial \alpha} \odot \alpha. \quad (10)$$

Given a simple example, we consider a sequential structure of supernet (i.e.  $f(x) = \sum_{o=1}^{|\mathcal{O}|} \alpha_{L,o} W^{L,o} \dots \sum_{o=1}^{|\mathcal{O}|} \alpha_{1,o} W^{1,o} x$ ), and we can find that  $\mathcal{F}_{\text{SF}}(\alpha)$  also yields the positive scores as SynFlow:

$$\mathcal{F}_{\text{SF}}(\alpha) = \left[ \mathbf{1}^T \left( \prod_{l=1}^L \sum_{o=1}^{|\mathcal{O}|} |\alpha_{l,o} * W^{l,o}| \right) \right]_{|\alpha_{i,j}|} \left[ \left( \prod_{l=1}^{i-1} \sum_{o=1}^{|\mathcal{O}|} |\alpha_{l,o} * W^{l,o}| \right) \mathbf{1} \right]. \quad (11)$$

As described, the scores  $\mathcal{F}_{\text{SF}}(\alpha)$  are all positive, showing that  $\mathcal{F}_{\text{SF}}(\alpha)$  will exclude the **zero** operation in the search space since its gradient is always 0, which is also in line with the final discretization stage in DARTS [25].

After defining the score metrics, we directly operate the architecture pruning on the DARTS’s initialized supernet  $W$  and architecture  $\alpha$  without any training. The importance of all available operations can be calculated by one-shot with only a batch of data based on Eq.(7)(8)(10), and the final architecture is obtained through *argmax*. As a result, our approach, which we call *training free differentiable architecture search* (**FreeDARTS**), can be implemented easily.

## 4. Experiments

In the above, we reformulate the differentiable architecture search from a pruning-at-initialization perspective, and propose a simple framework **FreeDARTS** for the architecture search without any training. In this section, we conduct a series of experiments to verify the foundational question: *can we find high-quality architectures without any training through our FreeDARTS?* We consider two cases to

analyze the proposed framework, including on the benchmark datasets [12, 33, 44] and DARTS search space [25].

### 4.1. Experiments on benchmark datasets

**Reproducible comparison with existing works on NAS-Bench-201.** The results for FreeDARTS and the weight-sharing NAS baselines on the NAS-Bench-201 set are provided in Table 1. FreeDARTS produced competitive results on all three datasets, significantly outperforming the DARTS and other elaborately designed methods. Moreover, the best single-run of FreeDARTS achieves a performance of **93.66%** on CIFAR-10, **70.78%** on CIFAR-100, and **46.53%** on ImageNet, which are very close to the optimal test accuracies in the NAS-Bench-201 dataset. The second block in Table 1 contains the comparison results of FreeDARTS with three existing train-free NAS methods. The Random baseline is to randomly generate architectures without training. NASWOT uses the Jacobian to score architectures, and TE-NAS uses the spectrum of NTKs and the number of linear regions to rank the architectures. Zero-cost NAS, zero-cots-pt NAS, and our FreeDARTS all consider Synflow metric in this experiment, where our FreeDARTS obtains competitive results, especially on the large ImageNet dataset. As shown, FreeDARTS outperforms the random baseline with a large margin, showing the effectiveness of FreeDARTS. Moreover, compared with the two elaborately designed train-free NAS, NASWOT and TE-NAS, FreeDARTS also achieves more competitive results, further showing the **reliability** of the proposed operation saliency metric. Table 1 also summarizes the search cost of several weight-sharing NAS baselines and training-free NAS methods. As shown, a significant advantage of our FreeDARTS is the **efficiency**, which only

Table 2. Zero-cost NAS and FreeDARTS with different saliency metrics on NAS-Bench-201.

Method	CIFAR-10		CIFAR-100		ImageNet-16-120	
	Valid(%)	Test(%)	Valid(%)	Test(%)	Valid(%)	Test(%)
Zero-cost with SNIP	89.08±0.96	91.82±1.56	68.00±2.16	68.08±2.01	37.29±5.91	37.12±5.88
Zero-cost with Grasp	88.10±0.65	90.92±0.81	66.26±1.40	66.35±1.17	34.66±4.83	33.88±4.43
Zero-cost with SynFlow	90.19±0.66	93.45±0.28	70.55±1.61	70.73±1.36	43.24±2.52	43.64±2.42
FreeDARTS with SNIP	89.57±0.57	92.96±0.52	69.77±0.76	69.90±0.80	42.66±1.51	43.79±1.54
FreeDARTS with Grasp	90.02±0.31	93.22±0.30	70.54±0.67	70.52±0.58	44.41±1.08	44.80±1.37
FreeDARTS with SynFlow	<b>90.39±0.30</b>	<b>93.52±0.11</b>	<b>70.96±0.51</b>	<b>70.78±0.08</b>	<b>44.68±1.67</b>	<b>45.28±1.73</b>

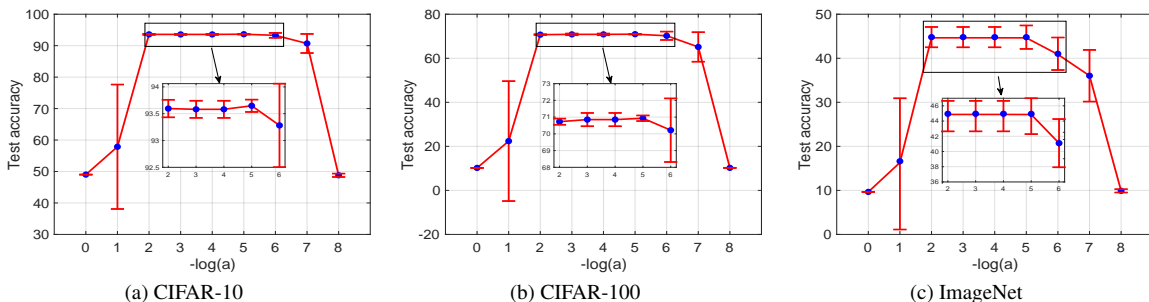


Figure 1. Hyperparameter analysis of FreeDARTS on the NAS-Bench-201 benchmark dataset.

Table 3. Statistic search results (test error) on NAS-Bench-1shot1.

Method	CIFAR-10 Average (%)		CIFAR-10 Best (%)		Search Cost
	Valid	Test	Valid	Test	
GDAS	6.8±0.1	6.1±0.2	6.7	5.9	11425s
PC-DARTS	6.7±0.1	6.2±0.2	6.6	5.9	14760s
DARTS (1st)	6.8±0.05	6.1±0.2	6.6	5.9	8280s
DARTS (2nd)	6.8±0.05	6.2±0.05	6.6	6.2	19800s
Random	24.4±32.8	24.1±33.2	7.8	7.5	N/A
FreeDARTS	7.8±2.4	7.3±2.4	<b>6.0</b>	<b>5.3</b>	<b>1.1s</b>

costs **0.6s** to find competitive architectures. More interesting, we can see train-free methods all cost much less time while achieved more competitive results, suggesting that the supernet training may be unnecessarily required in NAS.

**Comparison with different saliency metrics.** In our FreeDARTS, we consider two label-dependent saliency metrics, SNIP and GraSP, and a label-agnostic metric, Synflow, to measure the operation importance. Similarly, rather than calculating the saliency scores for  $\alpha$ , zero-cost NAS [1] also uses the three proxies to score an architecture, through summing scores of all weights  $\theta$  in the architecture. We compared the three saliency metrics for NAS, from  $\alpha$  level to  $\theta$  level. In Table 2, zero-cost NAS follows NASWOT [27] to randomly sample 100 architectures, and selects the best one based on the three saliency metrics. As shown in Table 2, without any training, the zero-cost NAS and FreeDARTS with different saliency metrics all achieved competitive results on NAS-Bench-201, outperforming most baselines in

Table 1. These results suggest that pruning at initialization is a promising direction for neural architecture search, which extremely improves the efficiency for NAS with obtaining competitive results. As shown in Table 2, our FreeDARTS framework generally outperforms zero-cost NAS under different saliency metrics, suggesting that evaluating architectures based on saliency metrics from operation level  $\alpha$  is more **reliable** than from supernet weights level  $\theta$ . Moreover, we can see from Table 2 that, the label-agnostic SynFlow based metric outperforms the remaining two label-dependent metrics both for zero-cost NAS and our FreeDARTS, showing labels maybe not necessarily required in the pruning at initialization based NAS. In the following experiments, we only consider the SynFlow based FreeDARTS to conduct hyperparameter studies and architecture search on the DARTS search space. Overall, the experiments in NAS-Bench-201 answered the foundational question, showing it is possible to find promising architectures without any training nor labels.

**Hyperparameter study.** As described in Sec.3.2, FreeDARTS is train-free, making our method is simple and concise to implement without tuning too many hyperparameters. As described in Sec. 3.2, the operation saliency score is the product of the value of  $\alpha$  and the gradient of  $\alpha$ , while  $\alpha$  is transformed by softmax before conduct the forward to calculate the gradient. Generally, the  $\alpha$  is initialized with  $a * randn$ , where  $a$  is weighted scale which is the only hyperparameter in our FreeDARTS denoting the trade-off between  $\frac{\partial \mathcal{R}_{FE}}{\partial \alpha}$  (or  $\frac{\partial \mathcal{L}}{\partial \alpha}$ ) and  $\alpha$ . Figure 1 analyzes the hyperparameter  $a$

Table 4. Comparison results with state-of-the-art NAS approaches on DARTS search space.

Method	Test Error (%)			Param (M)	+× (M)	Train Free	Label Agnostic	Search Cost (GPU day or sec.)
	CIFAR-10	CIFAR-100	ImageNet					
PARSEC [4]	2.86±0.06	-	26.3	3.6	509	×	×	0.6d
SNAS [39]	2.85±0.02	20.09	27.3 / 9.2	2.8	474	×	×	1.5d
BayesNAS [49]	2.81±0.04	-	26.5 / 8.9	3.4	-	×	×	0.2d
MdeNAS [48]	2.55	17.61	25.5 / 7.9	3.6	506	×	×	0.16d
GDAS [11]	2.93	18.38	26.0 / 8.5	3.4	545	×	×	0.2d
PDARTS [8]	2.50	16.63	24.4 / 7.4	3.4	557	×	×	0.3d
PC-DARTS [40]	2.57±0.07	17.11	25.1 / 7.8	3.6	586	×	×	0.3d
DrNAS [7]	2.54±0.03	16.30	24.2 / 7.3	4.0	644	×	×	0.4d
DARTS (1st) [25]	2.94	17.76	-	2.9	513	×	×	1.5d
DARTS (2nd) [25]	2.76±0.09	17.54	26.9 / 8.7	3.4	574	×	×	4d
TE-NAS [5]	2.63	17.83	26.2 / 8.3	3.8	610	✓	×	0.17d
Zero-Cost-PT [38]	2.68±0.17	17.53	24.4 / 7.5	4.7	817	✓	✓	0.018d
FreeDARTS	2.78±0.06	18.03	26.1 / 8.2	3.6	634	✓	✓	<b>1.5s</b>
FreeDARTS <sup>†</sup>	<b>2.50±0.05</b>	17.08	25.4 / 7.8	3.6	577	✓	✓	<b>1.5s</b>
FreeDARTS <sup>‡</sup>	2.67±0.04	<b>16.35</b>	<b>24.4 / 7.3</b>	4.1	655	✓	✓	<b>1.5s</b>

“Param” is the model size on CIFAR-10, while “+×” is calculated on ImageNet dataset. “d” is the GPU days and “s” is the GPU seconds. We only consider the Synflow based metric, which is label agnostic, for our FreeDARTS in this search space.

with summarizing the performance of FreeDARTS with different  $a$  on the NAS-Bench-201. In general, our FreeDARTS is robust to this hyperparameter, which with different  $a$  in a larger range ( $1e^{-5} \sim 1e^{-2}$ ) all achieve competitive results.

**Experimental results on NAS-Bench-1shot1.** We also conduct the experiments on the NAS-Bench-1shot1 space, where the comparison results for FreeDARTS and the weight-sharing NAS baselines are provided in Table 3. We report not only the average results but also the best results after several independent runs with different random seeds. As show, our FreeDARTS outperforms the Random baseline by large margins, showing the effectiveness of the proposed framework. As verified before, the most attractive advantage of our FreeDARTS is the efficiency, and it also completes the architecture search NAS-Bench-1shot1 space within much less time compared with the common differentiable NAS baselines, with only **1.1s**. Although these differentiable NAS baselines achieve better results according to the average test error, our FreeDARTS could find more competitive architectures based on the best test error. We need to notice that the ability to find the superior architecture is the core in the NAS compared with obtaining stable results. In addition, all differentiable NAS baselines achieve very similar results, and one possible reason is that the differentiable NAS is very easy to be tracked in the local optimal in this space.

## 4.2. Experiments on DARTS search space

**Search results on DARTS space.** We follow most existing NAS methods [25, 40], to conduct the architecture search on the CIFAR-10 dataset in this subsection, where the best-

found cell is repeatedly stacked to form the full structure for evaluation on CIFAR-10, CIFAR-100, and ImageNet datasets. We conducted the architecture search with different random seeds to obtain the architectures, and retrained them to select the best architecture based on the retrained validation performance. All experimental settings on our FreeDARTS are exact same as DARTS, while FreeDARTS<sup>†</sup> stacks 20 cells to form the backbone to eliminate the *depth gap*, and FreeDARTS<sup>‡</sup> directly searches on the ImageNet.

The comparison results with the state-of-the-art NAS methods are presented in Table 4. As shown, the best architecture searched by our FreeDARTS achieves a  $2.78 \pm 0.06$  % test error on CIFAR-10, which is on par with the DARTS while only costs 1.5 seconds. More interesting, after eliminate the *depth gap* by our FreeDARTS<sup>†</sup>, we achieve much better results with a 2.45 % test error on CIFAR-10, which outperforms the DARTS baseline by a large margin, again demonstrating the effectiveness of the proposed method. When transferring to larger datasets, the best architecture searched by FreeDARTS<sup>†</sup> also shows the competitive transferability, with 17.08 % test error on the CIFAR-100, and 25.4 / 7.8 % top1 / top5 test error on the ImageNet. When directly searching on the ImageNet, FreeDARTS<sup>‡</sup> obtains 16.35% test error on the CIFAR-100, and 24.8 / 7.5 % top1 / top5 test error on the ImageNet. Our FreeDARTS is also extremely efficient in DARTS space which completes the architecture search with only **1.5s**, while all existing NAS methods cost GPU days or hours.

**Comparison results on different search spaces.** As described above, due to the efficiency of our FreeDARTS, we

Table 5. Search results on with different settings.

Method	CIFAR-10 Test Error (%)		Param (M)
	Best	Mean	
GDAS	2.93	3.22±0.31	2.83±0.07
DARTS (1st)	2.94	3.22±0.45	2.02±0.41
DARTS (2nd)	2.62	3.02±0.26	2.83±0.07
FreeDARTS	2.75	2.92±0.18	3.82±0.26
FreeDARTS <sup>†</sup>	2.45	2.78±0.28	3.49±0.13
FreeDARTS <sup>‡</sup>	2.63	2.91±0.31	3.89±0.23

We report the best and mean test error after several searches. “Param” is the average model size and after several searches. “Mean” is the average test error of searched architectures.

can now perform the architecture search more flexibly, e.g., directly search with a larger backbone supernet with more cells to eliminate the *depth gap*, or directly on ImageNet dataset. In this subsection, we present an ablation study on the architecture search in different space. The comparison results are provided in Table 4 and 5, and all searched architectures with different *random seeds* are visualized in the Appendix. As shown, compared with the DARTS baselines which prefer those non-parameters operations, e.g. *skip and pooling*, the architectures search by our FreeDARTS, FreeDARTS<sup>†</sup>, and FreeDARTS<sup>‡</sup> all contain more parameters, implying that our FreeDARTS is more able to find valid and competitive architectures. From Table 5, we can also see that, the architectures searched by FreeDARTS<sup>†</sup> are generally better than those architectures searched on CIFAR-10 with smaller backbone, in terms of both the best and the average performance after several independent runs, showing the *depth gap* is a serious issue to be fixed. More interesting, although our FreeDARTS<sup>‡</sup> has a minor performance drop on CIFAR-10 compared with FreeDARTS<sup>†</sup>, it achieves higher evaluation performance on CIFAR-100 and ImageNet as it directly searched on ImageNet. The results in Table 4 and 5 verify that, our FreeDARTS can eliminate the *gap* between the architecture search and evaluation as it can **flexibly** search on different space, leading to more competitive results from our FreeDARTS<sup>†</sup> and FreeDARTS<sup>‡</sup>.

## 5. Discussion

This section discusses the advantages and the limitations of the proposed FreeDARTS framework by summarizing the above experimental findings. An attractive advantage of our FreeDARTS is the efficiency and effectiveness, which completes the entire architecture search in seconds with obtaining competitive results. The experimental results verified that the synaptic saliency based score metrics were not only effective in the network pruning but also in the architecture pruning. Since our FreeDARTS abandoned the supernet training phase, the common issue in differentiable NAS, that those parameter-free operations (e.g., *pooling* and *skip-*

*connection*, etc.) are preferred due to they are easy to be trained with producing more consistent outputs [40], is relieved. In addition, thanks to the memory efficiency by our FreeDARTS, we can eliminate the *depth gap* that we can conduct architecture search and evaluation in the same space. More important, our FreeDARTS is hyperparameter-efficient with only one hyperparameter to be tuned.

Apart from our FreeDARTS, several recent works [5, 27] also share the similar motivation as us to identify promising architectures at initialization. However, comparing with elaborately designing score functions, introducing the saliency metrics from a network pruning perspective is a more promising direction in train-free NAS, where FreeDARTS achieved more competitive results than NASWOT [27] and TE-NAS [5] with much less computational time.

FreeDARTS is a simple and direct application of network pruning at initialization to differentiable neural architecture search without any elaborately considerations, while it could still achieve more reliable results than differentiable NAS baselines with much less memory and computational consumption. This also implies that explicitly designing a more specific and explainable score function for NAS from the pruning perspective is a promising direction. In addition, our FreeDARTS simply prune the inferior operations by one-shot, while the iterative method may further enhance the performance as suggested in the network pruning [35].

## 6. Conclusion

This paper challenges the common practice in neural architecture search by raising the question: *can we measure the operation importance and find high-quality architectures without any training under the differentiable paradigm?* We approach this question by reformulating differentiable architecture search from a network pruning-at-initialization perspective, and introducing the saliency metrics to score the importance of operations in the supernet. Built based on saliency metrics, we propose a novel framework called *training free neural architecture search (FreeDARTS)*, which can perform architecture search extremely efficiently and more flexibly, with achieving competitive or even better results against SOTA in a training-free and even label-agnostic manner. Extensive experimental results on NAS benchmark dataset and the common DARTS search space verified the effectiveness of the proposed framework, showing the supernet training and labels are not necessarily required in differentiable NAS. We bridge the gap between the network pruning at initialization and the differentiable architecture search, and we hypothesize the information flow is a more appropriate indicator to reveal a good architecture than the validation performance in differentiable architecture search. We hope our findings encourage the community to further explore NAS from the perspective of network pruning at initialization.



## References

- [1] Mohamed S Abdelfattah, Abhinav Mehrotra, Łukasz Dudziak, and Nicholas D Lane. Zero-cost proxies for lightweight nas. In *ICLR*, 2021. 2, 3, 4, 5, 6
- [2] Bowen Baker, Otkrist Gupta, Ramesh Raskar, and Nikhil Naik. Accelerating neural architecture search using performance prediction. *arXiv preprint arXiv:1705.10823*, 2017. 2
- [3] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In *International Conference on Machine Learning*, pages 549–558, 2018. 1
- [4] Francesco Paolo Casale, Jonathan Gordon, and Nicolo Fusi. Probabilistic neural architecture search. *arXiv preprint arXiv:1902.05116*, 2019. 7
- [5] Wuyang Chen, Xinyu Gong, and Zhangyang Wang. Neural architecture search on imagenet in four gpu hours: A theoretically inspired perspective. In *ICLR*, 2021. 2, 3, 5, 7, 8
- [6] Xiangning Chen and Cho-Jui Hsieh. Stabilizing differentiable architecture search via perturbation-based regularization. In *ICML*, 2020. 1, 2
- [7] Xiangning Chen, Ruochen Wang, Minhao Cheng, Xiaocheng Tang, and Cho-Jui Hsieh. Drnas: Dirichlet neural architecture search. *arXiv preprint arXiv:2006.10355*, 2020. 7
- [8] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1294–1303, 2019. 1, 3, 7
- [9] Xuelian Cheng, Yiran Zhong, Mehrtash Harandi, Yunchao Dai, Xiaojun Chang, Tom Drummond, Hongdong Li, and Zongyuan Ge. Hierarchical Neural Architecture Search for Deep Stereo Matching. In *NeurIPS*, 2020. 1
- [10] Xuanyi Dong and Yi Yang. One-shot neural architecture search via self-evaluated template network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3681–3690, 2019. 5
- [11] Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. In *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 2019. 5, 7, 12
- [12] Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. *ICLR*, 2020. 5, 11
- [13] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019. 2
- [14] Song Han, Jeff Pool, John Tran, and William J Dally. Learning both weights and connections for efficient neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1*, pages 1135–1143, 2015. 3
- [15] Yiming Hu, Yuding Liang, Zichao Guo, Ruosi Wan, Xiangyu Zhang, Yichen Wei, Qingyi Gu, and Jian Sun. Angle-based search space shrinking for neural architecture search. In *European Conference on Computer Vision*, pages 119–134. Springer, 2020. 3
- [16] Steven A Janowsky. Pruning versus clipping in neural networks. *Physical Review A*, 39(12):6600, 1989. 3
- [17] Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabas Poczos, and Eric P Xing. Neural architecture search with bayesian optimisation and optimal transport. In *Advances in Neural Information Processing Systems*, pages 2020–2029, 2018. 1
- [18] Namhoon Lee, Thalaiyasingam Ajanthan, Stephen Gould, and Philip HS Torr. A signal propagation perspective for pruning neural networks at initialization. In *International Conference on Learning Representations*, 2020. 2, 3
- [19] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. Snip: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations*, 2019. 2, 3, 4
- [20] Changlin Li, Jiefeng Peng, Liuchun Yuan, Guangrun Wang, Xiaodan Liang, Liang Lin, and Xiaojun Chang. Block-wisely supervised neural architecture search with knowledge distillation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 1986–1995, 2020. 1
- [21] Changlin Li, Tao Tang, Guangrun Wang, Jiefeng Peng, Bing Wang, Xiaodan Liang, and Xiaojun Chang. Bossnas: Exploring hybrid cnn-transformers with block-wisely self-supervised neural architecture search. *arXiv preprint arXiv:2103.12424*, 2021. 1
- [22] Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. *arXiv preprint arXiv:1902.07638*, 2019. 2
- [23] Hanwen Liang, Shifeng Zhang, Jiacheng Sun, Xingqiu He, Weiran Huang, Kechen Zhuang, and Zhenguo Li. Darts+: Improved differentiable architecture search with early stopping. *arXiv preprint arXiv:1909.06035*, 2019. 1, 3
- [24] Chenxi Liu, Piotr Dollár, Kaiming He, Ross Girshick, Alan Yuille, and Saining Xie. Are labels necessary for neural architecture search? In *European Conference on Computer Vision*, pages 798–813. Springer, 2020. 1, 3
- [25] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *ICLR*, 2019. 1, 2, 3, 5, 7, 11, 12
- [26] Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l<sub>0</sub> regularization. In *International Conference on Learning Representations*, 2018. 3
- [27] Joseph Mellor, Jack Turner, Amos Storkey, and Elliot J Crowley. Neural architecture search without training. In *ICML*, 2021. 2, 3, 5, 6, 8
- [28] Asaf Noy, Niv Nayman, Tal Ridnik, Nadav Zamir, Sivan Doveh, Itamar Friedman, Raja Giryes, and Lihi Zelnik. Asap: Architecture search, anneal and prune. In *International Conference on Artificial Intelligence and Statistics*, pages 493–503. PMLR, 2020. 3
- [29] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameter shar-

- ing. In *International Conference on Machine Learning*, pages 4092–4101, 2018. [2](#)
- [30] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. *AAAI*, 2019. [1](#)
- [31] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A comprehensive survey of neural architecture search: Challenges and solutions. *arXiv preprint arXiv:2006.02903*, 2020. [1](#)
- [32] Christian Sciuto, Kaicheng Yu, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. Evaluating the search phase of neural architecture search. *arXiv preprint arXiv:1902.08142*, 2019. [1, 2](#)
- [33] Julien Siems, Lucas Zimmer, Arber Zela, Jovita Lukasiak, Margret Keuper, and Frank Hutter. Nas-bench-301 and the case for surrogate benchmarks for neural architecture search. *arXiv preprint arXiv:2008.09777*, 2020. [5, 11](#)
- [34] Suraj Srinivas and R Venkatesh Babu. Generalized dropout. *arXiv preprint arXiv:1611.06791*, 2016. [3](#)
- [35] Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. *Advances in Neural Information Processing Systems*, 33, 2020. [2, 3, 4, 8](#)
- [36] Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations*, 2020. [2, 3, 4](#)
- [37] Ruochen Wang, Minhao Cheng, Xiangning Chen, Xiaocheng Tang, and Cho-Jui Hsieh. Rethinking architecture selection in differentiable nas. In *ICLR*, 2021. [1, 2, 3, 4](#)
- [38] Lichuan Xiang, Lukasz Dudziak, Mohamed S Abdelfattah, Thomas Chau, Nicholas D Lane, and Hongkai Wen. Zero-cost proxies meet differentiable architecture search. *arXiv preprint arXiv:2106.06799*, 2021. [3, 5, 7](#)
- [39] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: stochastic neural architecture search. *ICLR*, 2019. [5, 7](#)
- [40] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. Pc-darts: Partial channel connections for memory-efficient architecture search. In *ICLR*, 2020. [5, 7, 8, 12](#)
- [41] Shen Yan, Yu Zheng, Wei Ao, Xiao Zeng, and Mi Zhang. Does unsupervised architecture representation learning help neural architecture search? *Advances in Neural Information Processing Systems*, 33, 2020. [1, 3](#)
- [42] Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. Nas-bench-101: Towards reproducible neural architecture search. In *ICML*, pages 7105–7114, 2019. [11](#)
- [43] Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marakchi, Thomas Brox, and Frank Hutter. Understanding and robustifying differentiable architecture search. In *ICLR*, 2020. [1, 2](#)
- [44] Arber Zela, Julien Siems, and Frank Hutter. Nas-bench-1shot1: Benchmarking and dissecting one-shot neural architecture search. In *ICLR*, 2020. [5, 11](#)
- [45] Chris Zhang, Mengye Ren, and Raquel Urtasun. Graph hypernetworks for neural architecture search. In *7th International Conference on Learning Representations*, 2019. [1, 2](#)
- [46] Miao Zhang, Huiqi Li, Shirui Pan, Xiaojun Chang, and Steven Su. Overcoming multi-model forgetting in one-shot nas with diversity maximization. In *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. [1](#)
- [47] Xuanyang Zhang, Pengfei Hou, Xiangyu Zhang, and Jian Sun. Neural architecture search with random labels. *arXiv preprint arXiv:2101.11834*, 2021. [1, 3](#)
- [48] Xiawu Zheng, Rongrong Ji, Lang Tang, Baochang Zhang, Jianzhuang Liu, and Qi Tian. Multinomial distribution learning for effective neural architecture search. In *International Conference on Computer Vision (ICCV)*, 2019. [7](#)
- [49] Hongpeng Zhou, Minghao Yang, Jun Wang, and Wei Pan. Bayesnas: A bayesian approach for neural architecture search. In *International Conference on Machine Learning*, pages 7603–7613, 2019. [7](#)
- [50] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018. [1](#)

## APPENDIX:

### A. Search space description and experimental setting

In our experiments, we consider two scenarios, NAS benchmark datasets, including NAS-Bench-101, NAS-Bench-1shot1, NAS-Bench-201, and NAS-Bench-301 [12, 33, 42, 44], and the common DARTS space [25], to analyze the proposed framework FreeDARTS. The search spaces of NAS-Bench-101, NAS-Bench-1shot1, and NAS-Bench-201 are much smaller than the common DARTS space, while the ground-truth for all candidate architectures in the benchmark datasets is known. The NAS-Bench-301 shares the same search space with DARTS space, while the performance of candidate architectures are obtained by a predictor fitted with  $\sim 60k$  architecture ground-truths.

The search space in NAS-Bench-201 [12] 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 [42] 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 [44] 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 to analyse FreeDARTS, 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  $\alpha_{normal}$  and a reduction cell  $\alpha_{reduce}$ . Cell structures are repeatedly stacked to form the final CNN structure. There are seven nodes in each cell: two input nodes, four operation nodes, and one output node. Each input node will select one operation from  $|\mathcal{O}| = 8$  candidate operations, including:  $3 \times 3$  max pooling and average pooling operation,  $3 \times 3$  and  $5 \times 5$  separable convolution operation,  $3 \times 3$  and  $5 \times 5$  dilated separable convolutions operation, identity, and *zero*. 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. We conduct the architecture search with 5 different *random seeds*, and the best one is selected after the evaluation on CIFAR-10. The best one

is then transferred to CIFAR-100 and ImageNet. Since the sizes of searched architectures are in a range, we adjust the number of filter in the evaluation to make the model sizes similar for fair comparison.

The NAS-Bench-301 [33] shares the same search space with DARTS, which contains  $10^{18}$  architectures, making it impossible to report the ground-truths for all architectures. Rather than training from the scratch to get the ground-truths for all architectures, NAS-Bench-301 fits a Graph Isomorphism Network based on the ground-truths of  $\sim 60k$  architecture to predict the performance of all remaining architectures. The prediction usually could hardly indicate the the true performance in practice. For example, the performance of an architecture containing all parameter-free operations still receive competitive predictive performance, in contrast to the extremely poor true performance. However, the authors showed that the prediction shows a positive correlation with the ground truth, that the resulting search trajectories by the prediction closely resemble the ground truth trajectories when evaluating a differentiable NAS method.

### B. Operation-level score and architecture-level score comparison

In Section 2 and Table 2, we have compared zero-cost NAS and FreeDARTS with the three different saliency metrics. We should notice that the major different between zero-cost NAS and FreeDARTS is that zero-cost NAS calculates the metrics from the whole architecture weights  $\theta$  level while FreeDARTS directly calculates the metrics on  $\alpha$ . Since zero-cost NAS sums scores of all parameters  $\theta$  in a specific architecture, it could only sample a set of architectures to calculate the scores for architectures rather than obtaining the scores of candidate operations for pruning. Similar to NAS-WOT, the sample size affects the performance of zero-cost NAS if we directly use it for architecture search rather than *warmup*. On the contrary, since FreeDARTS only needs to calculate the scores for candidate operations for once, there is no sampling-evaluation-selection process, and it is thus much more efficient than zero-cost NAS. Figure 2 plots the test accuracy of zero-cost NAS with different sample sizes under the three train-free saliency metrics on NAS-Bench-201. In our experiments, we found the calculation time of SNIP is similar to SynFlow, while GraSP is about ten times of them, since GraSP needs to calculate the second-order Hessian matrix (which is implemented by the Hessian-vector computation).

As shown, zero-cost NAS barely improves its performance with the sample sizes for SNIP and GraSP, implying the two metrics are not appropriate metrics to evaluate the whole architecture. On the contrary, the SynFlow, also adopted by our FreeDARTS, shows a clear improvement for zero-cost NAS with the sample size from 10 to 100, while it also decreases when the sample size increases to 1000. We

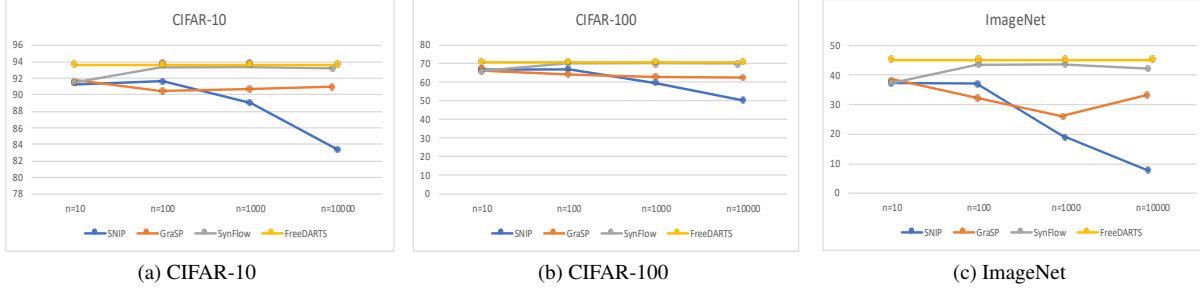


Figure 2. Test accuracy of FreeDARTS compared with zero-cost NAS with different sample sizes and saliency metrics on NAS-Bench-201.

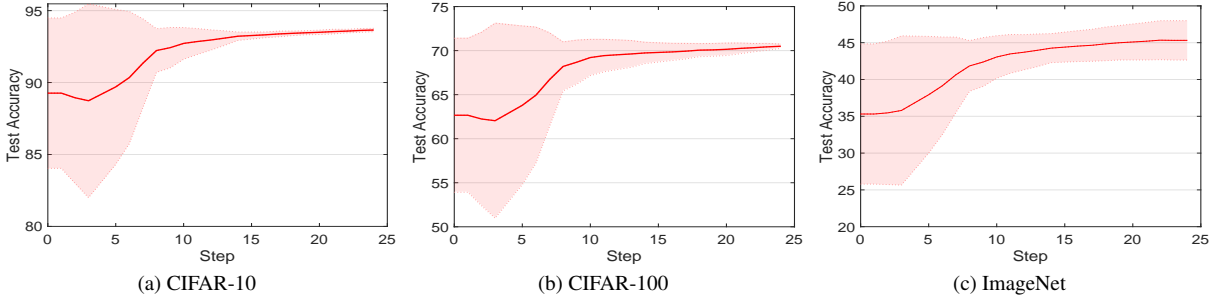


Figure 3. Track of test accuracy during the pruning for FreeNAS on the NAS-Bench-201.

can see a sharp drop for the zero-cost NAS with SNIP and GraSP when the sample size reaches 100 or more, showing that the architecture level-scoring is not reliable. On the contrary, the performance of our FreeDARTS would not vary since there is no need to sample a bunch of architectures for selection. We should also notice that the zero-cost NAS calculates the score for all  $\theta$  whose size is much larger than  $\alpha$ . In addition, zero-cost NAS needs to calculate the score for several hundred architectures, which means that the time cost of zero-cost NAS increases linearly with the sample size, resulting that zero-cost NAS is much more time-consuming than FreeDARTS.

### C. Analysis the operation remove by FreeDARTS

Algorithm 1 outlines the our FreeDARTS for the differentiable neural architecture search. As known, the DARTS utilized the trained magnitude of  $\alpha$  to indicate the importance of operations, while our FreeDARTS calculate the *operation saliency metrics* to indicate the importance of the operations. So, our FreeDARTS only replace the trained magnitude of  $\alpha$  with  $\mathcal{F}(\alpha)$ , and the discrete architecture can be obtained through apply *argmax* on  $\mathcal{F}(\alpha)$ . To investigate whether our FreeDARTS is effective in removing inferior operations, we consider only pruning one inferior operation from the supernet in each step based on the saliency metrics. Figure 3 tracks the quality of the pruned supernet, by applying the *argmax* on the architecture parameter  $\alpha$  of the pruned supernet after each step of operation pruning to get the architecture (Since  $\alpha$  is randomly generated, the *argmax* on the unpruned  $\alpha$  is

#### Algorithm 1 FreeDARTS

- 1: **input:** Initialized supernet weights  $W$  and architecture parameters  $\alpha$ ; Set of edges  $\mathcal{E}$  and set of candidate operations  $\mathcal{O}$ .
- 2: **for all** operations  $o \in \mathcal{O}_e$  **do**
- 3:     Calculate the operation saliency score for each operation  $\alpha_{e,o}$  based on Eq. (7), (8), or (10).
- 4: **end for**
- 5: Prune the candidate operation by one-shot based on  $\mathcal{F}_{\text{snip}}(\alpha_{e,o})$ ,  $\mathcal{F}_{\text{grasp}}(\alpha_{e,o})$ , or  $\mathcal{F}_{\text{SF}}(\alpha_{e,o})$ ;
- 6: **output:** Obtain a valid architecture  $\alpha^*$ .

Table 6. Statistic search results (test error) on NAS-Bench-301.

Method	Average	Best	Ground-True
GDAS [11]	6.52±0.62 (%)	5.38%	3.07±0.16 (%)
PC-DARTS [40]	6.42±0.43 (%)	5.46%	2.57±0.07 (%)
DARTS (2nd) [25]	6.74±0.58 (%)	5.87%	2.76±0.09 (%)
Random	7.11±0.58 (%)	6.21%	3.29±0.15 (%)
FreeDARTS(SNIP)	6.60±0.47 (%)	5.71%	2.69±0.08 (%)
FreeDARTS(GraSP)	6.72±0.48 (%)	5.74%	2.78±0.09 (%)
FreeDARTS(Synflow)	6.65±0.52 (%)	5.50%	2.50±0.05 (%)

same as random sampling when we do not remove inferior operations). As shown, the performance of the architectures increases with the pruning proceeding, verifying that the FreeDARTS can effectively remove inferior operations.



Table 7. Search results (test error) with the sample size on NAS-Bench-301.

Method	Average (10)	Best (10)	Average (100)	Best (100)	Average (1000)	Best (1000)
Random	7.11±0.58 (%)	6.21%	6.85±0.58 (%)	5.71%	6.89±0.55 (%)	5.61%
FreeDARTS(SNIP)	6.70±0.47 (%)	5.71%	6.77±0.51 (%)	5.65%	6.67±0.50 (%)	5.50%
FreeDARTS(GraSP)	6.72±0.48 (%)	5.74%	6.71±0.48 (%)	5.64%	6.65±0.49 (%)	5.55%
FreeDARTS(Synflow)	6.65±0.52 (%)	5.50%	6.66±0.53 (%)	5.49%	6.62±0.50 (%)	5.34%

## D. Experimental results on NAS-Bench-301

We also conduct the experiments on the NAS-Bench-301, the largest existing benchmark dataset, where the performance of our FreeDARTS with different operation saliency and tseveral differentiable NAS baselines are provided in Table 6. The ‘‘Average’’ reports the average predictive performance from the benchmark after several independent runs, and the ‘‘Best’’ reports the predictive performance of the best searched architectures. The ‘‘Ground-Truth’’ is the validation results based on the train-from-the-scratch. As show, our FreeDARTS also outperforms the Random baseline by large margins in the NAS-Bench-301, again showing the effectiveness of the proposed framework.

As we can observed, the predictive results in NAS-Bench-301 are not very consistent with the ground truth in Table 6. For example, although GDAS achieves much lower ground truth performance, it obtains the best predictive performance from the NAS-Bench-301. Although the predictive performance from NAS-Bench-301 could not exactly indicate the true performance, the results still present several consistent results with our previous observation in other benchmark datasets. For example, our FreeDARTS with different operation saliency metrics outperforms ‘‘Random’’ baseline by large margins, and the Synflow based metric achieved the best results among the three operation saliency metrics. Compared with several NAS baselines, our FreeDARTS without any training also achieves comparable results.

In addition, Table 7 shows how the performance of our FreeDARTS varies with different random seeds. We report the average and best performance of FreeDARTS under different number of seeds. The ‘‘Random’’ baseline also reports the average performance in NAS-Bench-301 with varying the number of seeds. We can see that the ‘‘Best’’ performance increase when we evaluate more randomly sampled points with different seeds, which makes sense as we randomly select architecture from the search space. In contrast, our FreeDARTS with different saliency metrics are more robust, where the best performance only changes slightly with different random seeds. As shown, compared with the random baseline, our FreeDARTS presents more robust results.

## E. Visualization of the searched architectures

Since the inputs of FreeDARTS with Synflow are all-ones vector, the search results will be the same for ar-

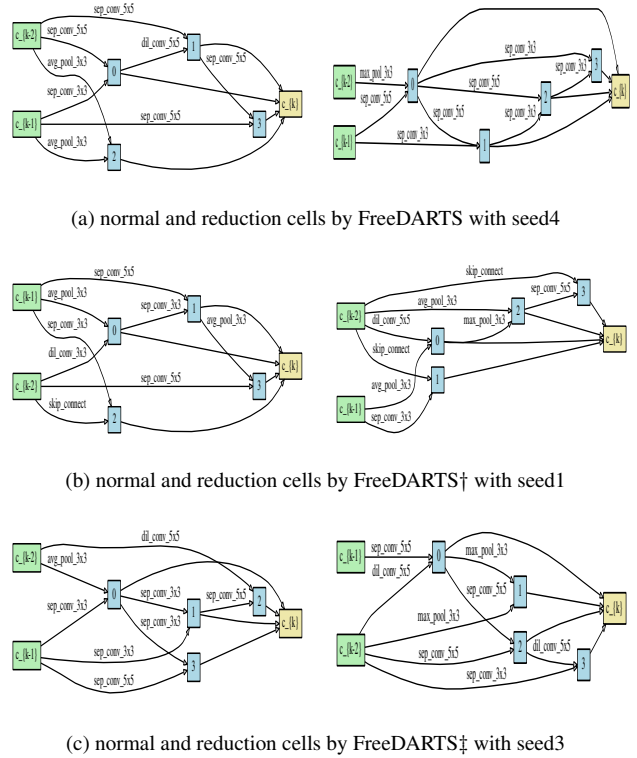
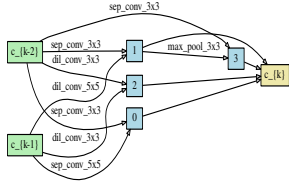


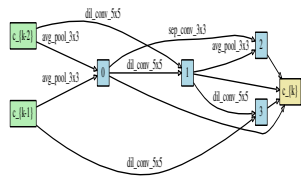
Figure 4. The best cells discovered by FreeDARTS, FreeDARTS†, and FreeDARTS‡(from top to bottom) on the DARTS search space.

chitecture search on CIFAR-10 and CIFAR-100. So, we only perform the architecture search on CIFAR-10 and ImageNet. In result, there are 15 architectures searched by our FreeDARTS, FreeDARTS†, and FreeDARTS‡ with 5 different random seeds 0-4. In Figure 4, we present the best architectures searched in DARTS space by FreeDARTS(seed4), FreeDARTS†(seed1), and FreeDARTS‡(seed3), and the remaining searched architectures are presented in Figure 5. An interesting finding is that, our FreeDARTS avoid selecting too many weight-free operations (e.g., pooling and skip-connection), that all searched architectures contains less than two weight-free operations in the normal cells <sup>1</sup>.

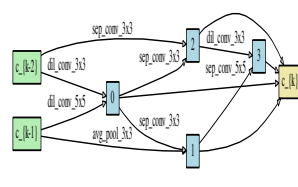
<sup>1</sup>The reproducible codes could be found in the supplementary material.



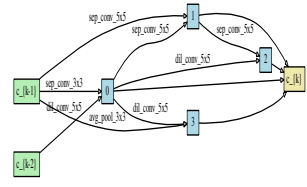
(a) normal and reduction cell by FreeDARTS with seed0



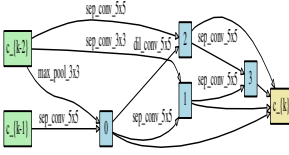
(b) normal and reduction cell by FreeDARTS with seed1



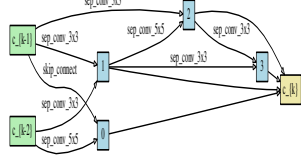
(c) normal and reduction cell by FreeDARTS with seed2



(d) normal and reduction cell by FreeDARTS with seed3



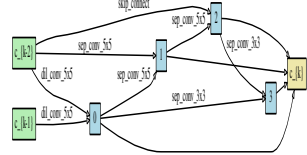
(e) normal and reduction cell by FreeDARTS† with seed0



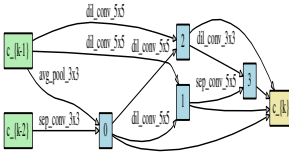
(f) normal and reduction cell by FreeDARTS† with seed2



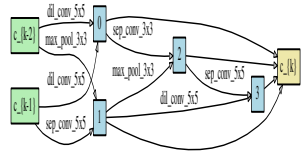
(g) normal and reduction cell by FreeDARTS† with seed3



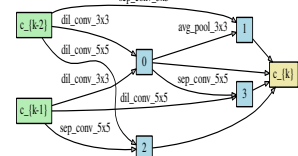
(h) normal and reduction cell by FreeDARTS† with seed4



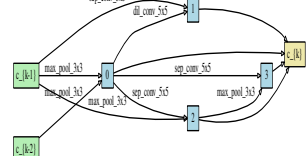
(i) normal and reduction cell by FreeDARTS‡ with seed0



(j) normal and reduction cell by FreeDARTS‡ with seed1



(k) normal and reduction cell by FreeDARTS‡ with seed2



(l) normal and reduction cell by FreeDARTS‡ with seed4

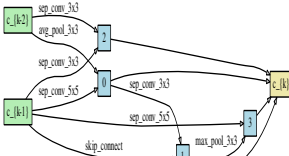


Figure 5. The cells discovered by FreeDARTS, FreeDARTS, and FreeDARTS‡ with different *random seeds* on the DARTS search space.