Dynamic Ensemble of Low-Fidelity Experts: Mitigating NAS "Cold-Start"

Junbo Zhao[*1,3], **Xuefei Ning**[*†1], **Enshu Liu**[1], **Binxin Ru**[4], **Zixuan Zhou**[1], **Tianchen Zhao**[1], **Chen Chen**[2], **Jiajin Zhang**[2], **Qingmin Liao**[3], **Yu Wang**[†1]

[1]Department of Electronic Engineering, Tsinghua University
[2]Huawei Technologies Co., Ltd
[3]Tsinghua Shenzhen International Graduate School
[4]SailYond Technology & Research Institute of Tsinghua University in Shenzhen
[†]foxdoraame@gmail.com, [†]yu-wang@tsinghua.edu.cn

Abstract

Predictor-based Neural Architecture Search (NAS) employs an architecture performance predictor to improve the sample efficiency. However, predictor-based NAS suffers from the severe "cold-start" problem, since a large amount of architecture-performance data is required to get a working predictor. In this paper, we focus on exploiting information in cheaper-to-obtain performance estimations (i.e., low-fidelity information) to mitigate the large data requirements of predictor training. Despite the intuitiveness of this idea, we observe that using inappropriate low-fidelity information even damages the prediction ability and different search spaces have different preferences for low-fidelity information types. To solve the problem and better fuse beneficial information provided by different types of low-fidelity information, we propose a novel dynamic ensemble predictor framework that comprises two steps. In the first step, we train different sub-predictors on different types of available low-fidelity information to extract beneficial knowledge as low-fidelity experts. In the second step, we learn a gating network to dynamically output a set of weighting coefficients conditioned on each input neural architecture, which will be used to combine the predictions of different low-fidelity experts in a weighted sum. The overall predictor is optimized on a small set of actual architecture-performance data to fuse the knowledge from different low-fidelity experts to make the final prediction. We conduct extensive experiments across five search spaces with different architecture encoders under various experimental settings. For example, our methods can improve the Kendall's Tau correlation coefficient between actual performance and predicted scores from 0.2549 to 0.7064 with only 25 actual architecture-performance data on NDS-ResNet. Our method can easily be incorporated into existing predictor-based NAS frameworks to discover better architectures. Our method will be implemented in Mindspore (Huawei 2020), and the example code is published at https://github.com/A-LinCui/DELE.

Introduction

In recent years, architectures automatically designed by neural architecture search (NAS) (Elsken, Metzen, and Hutter 2019) have achieved state-of-the-art performance on various tasks (Zoph and Le 2016; Liu, Simonyan, and Yang 2018; Chen et al. 2019; Wang et al. 2020). Accurate and efficient architecture performance estimation strategy is one of the key components of NAS (Elsken, Metzen, and Hutter 2019), which can be broadly divided into parameter-sharing-based (Pham et al. 2018) and predictor-based methods (Luo et al. 2018; Ning et al. 2020; White et al. 2021). The former evaluates with weights shared in an over-parametrized super network, while the latter learns a predictor to predict the performance of candidate architectures.

Predictor-based NAS trains an approximate performance predictor and utilizes it to rank unseen architectures without actually training them. Therefore, once we have a predictor that can reliably rank the performance of unseen architectures, the architecture exploration can be significantly accelerated. However, predictor-based NAS suffers from the severe "cold-start" problem: It usually takes quite a considerable cost to acquire the architecture-performance data needed for training a working predictor from scratch.

Recognizing the high cost of getting actual architecture-performance data as the major challenge for predictor-based NAS, existing efforts seek to learn the predictor in a more data-efficient way. Researchers have designed specialized predictor architectures (Ning et al. 2020; Zhang et al. 2019; Tang et al. 2020; Yan et al. 2021; Ning et al. 2022), training losses (Luo et al. 2018; Ning et al. 2020; Xu et al. 2021; Tang et al. 2020; Yan et al. 2020, 2021), to exploit information in the limited data more efficiently. In contrast, **our work focuses on exploiting more information in other cheaper-to-obtain performance estimations (i.e., low-fidelity information) to mitigate the data requirements of predictor training.** Actually, it is intuitive that utilizing other low-fidelity information (e.g., grasp (Wang, Zhang, and Grosse 2020) and plain (Mozer and Smolensky 1988)) for predictor training can help mitigate the cold-start problem. One can anticipate that training with this information might bring potential improvements in two aspects. On the one hand, the ranking information included in some indicators (e.g., one-shot (Pham et al. 2018), zero-shot (Abdelfattah et al. 2021a; Lin et al. 2021) estimations) might help the predictor acquire a better ranking quality. On the other hand, learning to fit other low-fidelity information could encourage the predictor

[*]These authors contributed equally.
[†]Corresponding authors.

to extract better architecture representations.

A straightforward way of utilizing low-fidelity information is to pretrain the model on a single type of low-fidelity information and finetune it on a small amount of actual architecture-performance data. We conduct a preliminary experiment in Table 1 and make the following observations.

- *Low-fidelity information does have the potential to improve prediction ability with limited actual architecture-performance data significantly. E.g., utilizing grad_norm (Abdelfattah et al. 2021b) increases the relative Kendall's Tau*[1] *for 0.1597 and 0.8286 on NAS-Bench-201 (Dong and Yang 2020) and NDS-ResNet (Radosavovic et al. 2019), respectively.*
- *Inappropriate low-fidelity information types even damage the prediction ability. E.g., utilizing "plain" decreases the relative Kendall's Tau for 0.1423, 0.7677 on NAS-Bench-201 and NAS-Bench-301 (Siems et al. 2020), respectively.*
- *Different search spaces have different preferences for low-fidelity information types. E.g., grad_norm decreases Kendall's Tau on NAS-Bench-301 but benefits the prediction on the other search spaces.*
- *A high-ranking quality of the low-fidelity information does not indicate its utilization effectiveness. E.g., synflow (Tanaka et al. 2020) positively correlates with actual performance but damages prediction on NAS-Bench-301.*

That is to say, despite the intuitiveness of this idea, which types of low-fidelity information are useful for performance prediction is unclear to practitioners beforehand. In addition, different types of low-fidelity information could provide beneficial information from different aspects, but the naive method described above can only utilize one type of low-fidelity information. Therefore, it would be better if we could fuse the knowledge from multiple types of low-fidelity information organically in an automated way.

In this paper, **we propose a novel dynamic ensemble predictor framework**, whose core is a learnable gating network that maps the neural architecture to a set of weighting coefficients to be used in ensembling predictions of different low-fidelity experts. The framework comprises two steps. In the first step, we pretrain different low-fidelity experts on different types of available low-fidelity information to extract beneficial knowledge. In the second step, the overall predictor is finetuned on the actual architecture-performance data to fuse knowledge from different types of low-fidelity information to make the final prediction. In this way, we can not only leverage multiple low-fidelity information in the architecture performance prediction but also balance their contributions in an automatic and dynamic fashion, overcoming the challenge for the practitioners to decide on which low-fidelity information to use.

To demonstrate the effectiveness of our proposed method, we conduct extensive experiments across multiple benchmarks, including NAS-Bench-201, NAS-Bench-301, NDS ResNet, NDS ResNeXt-A (Radosavovic et al. 2019), and MobileNetV3 (Cai et al. 2019). And our experiments are conducted under various experimental settings (*e.g.,* different predictor construction methods, varying data sizes). We show that our method of exploiting additional low-fidelity information can significantly and consistently improve the ranking quality of predictors compared to using the architecture-performance solely, thus improving the overall NAS efficiency. Our method can be easily incorporated into existing predictor-based NAS methods to alleviate the cold-start problem and guide the NAS process to discover better architectures. For example, our dynamic ensemble predictor discovers architectures with 94.37% test accuracy on NAS-Bench-201 (CIFAR-10 (Krizhevsky, Hinton et al. 2009)), surpassing ReNAS (Xu et al. 2021) (93.99%) and NEPNAS (Wei et al. 2020) (91.52%) with the same search budget. More information is available on our website https://sites.google.com/view/nas-nicsefc/home/search-strategy-improvement/dele.

[1]Kendall's Tau is the relative difference of the number of concordant pairs and discordant pairs, reflecting the ranking correlation between predictions and ground-truths.

Related Work

Fast Evaluation Strategies in NAS

Neural architecture search (NAS) (Elsken, Metzen, and Hutter 2019) is a technique to design neural network architectures automatically. The vanilla NAS method (Zoph and Le 2016) is computationally expensive since it needs to train each candidate architecture from scratch to get its performance. Therefore, a series of methods focus on developing faster architecture evaluation strategies to address the computational challenge. The two most popular types of fast evaluation strategies are the one-shot estimators (Bender et al. 2018; Pham et al. 2018; Guo et al. 2020) and zero-shot estimators (Mellor et al. 2020; Abdelfattah et al. 2021a).

One-shot performance estimations. One-shot NAS methods (Bender et al. 2018; Pham et al. 2018; Guo et al. 2020) construct an over-parametrized network (namely supernet), in which all candidate architectures are contained and share weights. After being trained to convergence, the supernet can evaluate the performance of each architecture by directly using the corresponding weights. Due to its efficiency, the one-shot performance estimation strategy is widely studied and used on different search spaces (Cai et al. 2019; Wu et al. 2019) and for different tasks (Chen et al. 2019; Wang et al. 2020). However, as reported in EEPE (Ning et al. 2021), one-shot performance estimations might have unsatisfying correlation and prominent bias. Therefore, one-shot performance estimation can fail to benefit NAS (Pourchot, Ducarouge, and Sigaud 2020).

Zero-shot performance estimations. Recently, several researches (Mellor et al. 2020; Abdelfattah et al. 2021a; Lin et al. 2021) propose "zero-shot" estimators, which utilize randomly initialized weights to estimate architectures' performance. Since no training process is required, these estimations are extremely fast. Nevertheless, EEPE (Ning et al. 2021) reveals that these zero-shot estimations have prominent biases, no zero-shot estimator can get a satisfying ranking quality in all search spaces, and the best zero-shot estimator is different across search spaces.

Low-fidelity Type	Low-fidelity Corr. / Kendall's Tau Relative Improvement				
	NAS-Bench-201	NAS-Bench-301	NDS-ResNet	NDS-ResNeXt-A	MobileNet-V3
grasp	+0.3227 / -0.0118	+0.4062 / +0.0189	-0.1142 / -0.9431	-0.2615 / -1.0230	-0.0663 / +0.0091
plain	-0.1467 / -0.1423	-0.4670 / -0.7677	+0.3066 / +0.7477	+0.2887 / +0.4064	+0.0116 / -0.0544
synflow	+0.5808 / +0.1463	+0.1967 / -0.2653	+0.2307 / +0.7270	+0.6904 / +1.0406	+0.6366 / -0.0024
grad_norm	+0.4798 / +0.1597	+0.0378 / -0.3340	+0.2372 / +0.8286	+0.3190 / +0.6247	+0.0696 / +0.0238
jacob_cov	+0.4763 / +0.1780	+0.0958 / -0.1654	-0.0724 / -0.5551	+0.0510 / -0.5681	-0.0053 / -0.1423

Table 1: The "Low-Fidelity Corr." and relative Kendall's Tau improvement achieved by utilizing different typical types of low-fidelity information.Specifically, we construct the predictor with an LSTM encoder and train it with ranking loss. All architectures in the training split are used for pretraining, while the first 1% percentages by index with corresponding actual performance are used for finetuning. "Low-Fidelity Corr." represents Kendall's Tau correlation between the low-fidelity information and the actual performance.

Predictor-based NAS

Predictor-based NAS (Luo et al. 2018; Ning et al. 2020; Wei et al. 2020; Tang et al. 2020; Xu et al. 2021; White et al. 2021; Ning et al. 2022) is another type of NAS methods that relies on an architecture performance predictor. An architecture performance predictor takes the architecture description as the input and outputs an estimated score. In each iteration of predictor-based NAS, the predictor is trained on actual architecture-performance data and then utilized to efficiently evaluate and sample new architectures. Then, the architecture-performance data of the newly sampled architectures would be used to tune the predictor in the next iteration. The most costly part of the predictor-based NAS flow is getting the actual architecture-performance data for predictor training. We refer the readers to the GATES paper (Ning et al. 2020) for a summary of the general predictor-based NAS workflow. Recently, Wu *etal.* (Wu et al. 2021) derives a formulation for predictor-based NAS and justify the rationality of this widely-used workflow.

A problem with predictor-based NAS is that we usually need many actual architecture-performance data to get a working predictor. The initial exploration in the search space is poorly guided and usually just conducted by random sampling. We refer to this problem as the "cold-start problem".

Improving Predictor-based NAS. Researchers have been focused on making the predictor utilize available data more efficiently. Existing methods can be resolved into two aspects: 1) **The construction of predictor architectures**: NASBot (Kandasamy et al. 2018) employs Gaussian Process as the predictor to better model the uncertainty. For topological search spaces, graph-based predictors are designed to encode the architecture in a better way (Ning et al. 2020; Dudziak et al. 2020; Shi et al. 2020; Ning et al. 2022). Tang *etal.* (Tang et al. 2020) propose explicitly modeling the relation between multiple architectures to predict their performances. 2) **The loss design of predictor training**: GATES (Ning et al. 2020) and ReNAS (Xu et al. 2021) propose to train the predictor with ranking loss to provide better architecture comparison. Several other studies (Luo et al. 2018; Tang et al. 2020; Yan et al. 2020) employ reconstruction loss as an auxiliary loss term.

Utilizing Cheaper-to-obtain Estimations. Recently, several studies have attempted to exploit cheaper-to-obtain performance estimations in predictor-based NAS to improve search efficiency: 1) ProxyBO (Shen et al. 2021) proposes to combine the architecture ranking given by the predictor and zero-shot proxies in the search process; 2) White *etal.* (White et al. 2021) find that certain families of performance estimations can be combined to achieve even better predictive power; 3) AceNAS (Zhang et al. 2021) proposes to pretrain the predictor with FLOPs, parameter size, and weight-sharing accuracy in a multi-task manner. However, these methods either heavily rely on the high-ranking correlation between the actual performance and the utilized estimations (Shen et al. 2021) or require carefully utilized estimation selection (White et al. 2021; Zhang et al. 2021). Different from these attempts, our method has no requirement for correlation between the utilized estimations and actual performance and can use a broader range of cheaper-to-obtain estimations without the need for manual hand-picking.

Low-Fidelity Information

Low-fidelity information refers to indicators obtained with a low computational cost. These indicators capture some properties of neural architectures and thus can indicate their performances to some extent. We anticipate that learning to fit these cheaper-to-obtain data can encourage the predictor to extract better architecture representations and thus boost the ranking quality of the predictor. Different types of low-fidelity information can be roughly classified as follows.

- **One-shot information.** The performance of architectures obtained from the one-shot supernet.
- **Zero-shot information**, such as grad_norm, synflow and synflow_bn (Tanaka et al. 2020), snip (Lee, Ajanthan, and Torr 2018), grasp, fisher (Theis et al. 2018) and jacob_cov (Mellor et al. 2020).
- **Complexity information.** Architecture information from the complexity perspective, such as the number of floating-point operations (FLOPs), the parameter size (params), and the inference latency (latency).

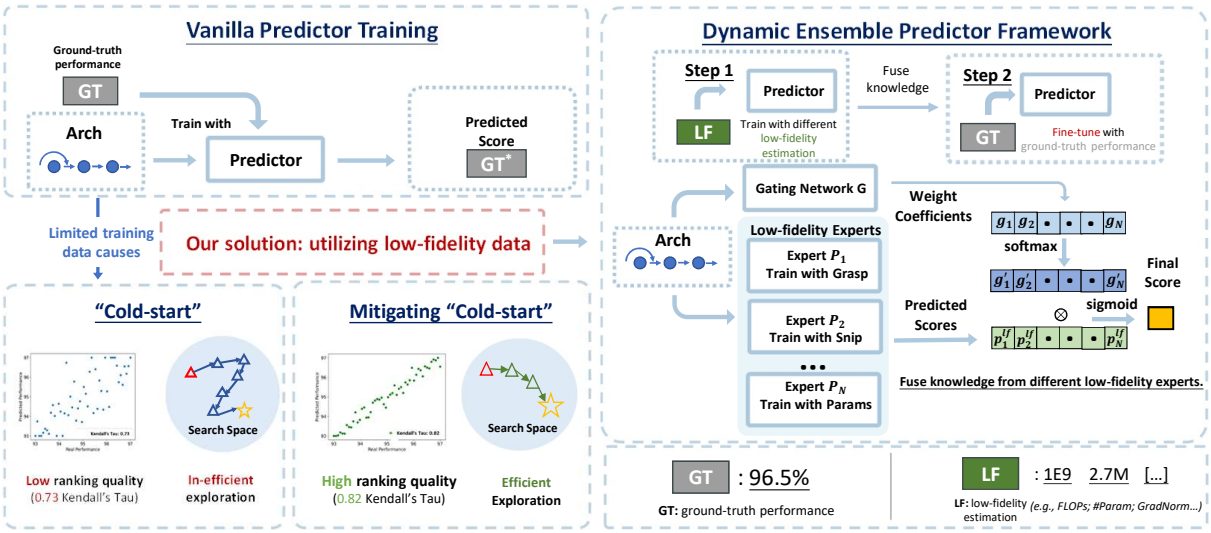


Figure 1: Illustration of our motivation and our proposed dynamic ensemble predictor framework.

The Proposed Method

In this section, we present the dynamic ensemble performance predictor framework. The illustration of our motivation and the predictor framework are shown in Figure 1.

Dynamic Ensemble Performance Predictor

Dynamic Ensemble Neural Predictor. Suppose we have N predictors $\{P_i\}_{i=1}^N$ (i.e., low-fidelity experts), each of which takes the architecture α as the input and outputs a predicted score. We represent the predicted scores for the architecture α as

$$p^{\text{lf}}(\alpha) = [p_1^{\text{lf}}(\alpha), p_2^{\text{lf}}(\alpha), \cdots, p_N^{\text{lf}}(\alpha)] \in \mathbb{R}^N \tag{1}$$

where $p_i^{\text{lf}}(\alpha)$ denotes the score predicted by the expert P_i.

We learn a gating network G to ensemble these experts to fuse beneficial knowledge from different types of low-fidelity information. The gating network dynamically maps each neural architecture to a set of weights, which are used as the weighting coefficients of predictions from different low-fidelity experts. This enables us to leverage multiple sources of low-fidelity information without worrying about which one is more relevant for the current search space. We utilize the same predictor architecture as the gating network architecture.

The final predicted score $p(\alpha)$ can be written as

$$k_i(\alpha) = p_i^{\text{lf}}(\alpha) \cdot \frac{\exp(g_i(\alpha))}{\sum_{j=1}^N \exp(g_j(\alpha))}$$

$$p(\alpha) = \text{sigmoid}(G(\alpha)^T p^{\text{lf}}(\alpha)) = \text{sigmoid}(\sum_{i=1}^N k_i(\alpha)) \tag{2}$$

where $g_i(\alpha)$ and $k_i(\alpha)$ denote the weighting coefficient and weighted score for the i^{th} low-fidelity expert, and $G(\alpha) \in \mathbb{R}^N$ denotes the weighted coefficient vector after softmax. The gating network can learn to tailor different weighting coefficients for different input architectures.

Training Framework. Our training process for the dynamic ensemble neural predictor consists of two steps. In the first step, we train a predictor on each type of low-fidelity information to extract expert knowledge, as formalized below:

$$w_i^* = \underset{w_i}{\text{argmin}}\, E_{x^{\text{lf}} \sim D_i^{\text{lf}}}[\mathcal{L}(x^{\text{lf}}, P_i(w_i))] \tag{3}$$

where $x^{\text{lf}} \sim D_i^{\text{lf}}$ denotes the data sampled from the training dataset of the i^{th} type of low-fidelity information and w_i denotes the weights of the i^{th} predictor. In the second step, we construct and finetune the entire predictor on the actual performance data,

$$\{w_i^*\}_{i=1}^N, w_g^* = \underset{\{w_i\}_{i=1}^N, w_g}{\text{argmin}}\, E_{x^{\text{tr}} \sim D^{\text{tr}}}[\mathcal{L}(x^{\text{tr}}, P(\{w_i\}_{i=1}^N, w_g))] \tag{4}$$

where w_g denotes the weights of the gating network; $x^{\text{tr}} \sim D^{\text{tr}}$ denotes the data sampled from the training dataset of actual performances.

Discussion about Simplicity. Our method is easily understandable and applicable. Firstly, our method requires no hyper-parameter tuning nor careful low-fidelity information selection. Secondly, our method is general to different search spaces, datasets and encoders, since its two core designs are general instead of specially designed for specific search space properties: 1) utilizing low-fidelity information to improve the prediction ability; 2) dynamically assembling low-fidelity experts to fuse beneficial knowledge from different low-fidelity information. Our method can be applied as long as several low-fidelity information (not necessarily having a good correlation with the actual performance) and an arbitrary architecture encoder are available for the search space.

Overall Search Flow

Our predictor-based flow goes as follows. In the first phase, we mitigate the cold-start issue by utilizing low-fidelity estimation. Specifically, we randomly sample N_0 architectures

Search Space	Encoder	Manner	Proportions of training samples				
			1%	5%	10%	50%	100%
NAS-Bench-201	GATES	Vanilla	$0.7332_{(0.0110)}$	$0.8582_{(0.0059)}$	$0.8865_{(0.0045)}$	$0.9180_{(0.0029)}$	$0.9249_{(0.0019)}$
		Ours	$\mathbf{0.8244}_{(0.0081)}$	$\mathbf{0.8948}_{(0.0021)}$	$\mathbf{0.9075}_{(0.0015)}$	$\mathbf{0.9216}_{(0.0019)}$	$\mathbf{0.9250}_{(0.0020)}$
	LSTM	Vanilla	$0.5692_{(0.0087)}$	$0.6410_{(0.0018)}$	$0.7258_{(0.0053)}$	$0.8765_{(0.0010)}$	$0.9000_{(0.0008)}$
		Ours	$\mathbf{0.7835}_{(0.0062)}$	$\mathbf{0.8538}_{(0.0029)}$	$\mathbf{0.8683}_{(0.0015)}$	$\mathbf{0.8992}_{(0.0010)}$	$\mathbf{0.9084}_{(0.0010)}$
NAS-Bench-301	GATES	Vanilla	$0.4160_{(0.0450)}$	$0.6752_{(0.0088)}$	$0.7354_{(0.0044)}$	$0.7693_{(0.0041)}$	$\mathbf{0.7883}_{(0.0011)}$
		Ours	$\mathbf{0.5529}_{(0.0135)}$	$\mathbf{0.6830}_{(0.0038)}$	$\mathbf{0.7433}_{(0.0018)}$	$\mathbf{0.7752}_{(0.0026)}$	$0.7842_{(0.0022)}$
	LSTM	Vanilla	$0.4757_{(0.0150)}$	$0.6116_{(0.0099)}$	$0.6923_{(0.0044)}$	$0.7516_{(0.0017)}$	$0.7667_{(0.0007)}$
		Ours	$\mathbf{0.4805}_{(0.0083)}$	$\mathbf{0.6405}_{(0.0035)}$	$\mathbf{0.7075}_{(0.0022)}$	$\mathbf{0.7544}_{(0.0028)}$	$\mathbf{0.7751}_{(0.0011)}$
NDS-ResNet	LSTM	Vanilla	$0.2549_{(0.0087)}$	$0.4564_{(0.0108)}$	$0.5770_{(0.0094)}$	$0.7758_{(0.0078)}$	$0.8244_{(0.0110)}$
		Ours	$\mathbf{0.7064}_{(0.0109)}$	$\mathbf{0.7548}_{(0.0080)}$	$\mathbf{0.7652}_{(0.0037)}$	$\mathbf{0.8271}_{(0.0054)}$	$\mathbf{0.8383}_{(0.0049)}$
NDS-ResNeXt-A	LSTM	Vanilla	$0.3568_{(0.0327)}$	$0.6243_{(0.0220)}$	$0.6671_{(0.0307)}$	$0.8224_{(0.0091)}$	$0.8701_{(0.0051)}$
		Ours	$\mathbf{0.7753}_{(0.0010)}$	$\mathbf{0.8276}_{(0.0024)}$	$\mathbf{0.8398}_{(0.0044)}$	$\mathbf{0.8453}_{(0.0040)}$	$\mathbf{0.8777}_{(0.0042)}$
MobileNet-V3	LSTM	Vanilla	$0.7373_{(0.0041)}$	$0.7852_{(0.0028)}$	$0.7832_{(0.0040)}$	$0.7944_{(0.0028)}$	$0.8023_{(0.0014)}$
		Ours	$\mathbf{0.7698}_{(0.0018)}$	$\mathbf{0.8034}_{(0.0027)}$	$\mathbf{0.8042}_{(0.0019)}$	$\mathbf{0.8084}_{(0.0017)}$	$\mathbf{0.8135}_{(0.0020)}$

Table 2: The Kendall's Tau (average over five runs) of using different encoders on NAS-Bench-201, NAS-Bench-301, NDS-ResNet, NDS-ResNeXt-A and MobileNet-V3. And the standard deviation is in the subscript. The detailed dataset split is elaborated in the appendix. "Vanilla" represents directly training predictor with ground-truth accuracies without low-fidelity information utilization.

from the search spaces for ground-truth performance and M for low-fidelity information evaluation, respectively. Next, these data are used to train an initial predictor with the dynamic ensemble method.

In the second phase, we run a predictor-based search for T_p stages. In each stage, a T_{pe}-step tournament-based evolutionary search (Real et al. 2019) (population size π, tournament μ) with scores evaluated by the predictor are run for N_p times. We query for rewards of the N_p sampled architectures and then finetune the predictor on all known architecture-performance data for K epochs. In total, we query the actual performance of architectures in search space for $N_0 + T_p \times N_p$ times. The test accuracy of the architecture with the highest reward among all sampled architectures is reported.

Note that our method is a predictor pretraining method that can be easily incorporated into most predictor-based NAS frameworks to alleviate the cold-start problem. It is compatible with different types of predictor architectures (Luo et al. 2018; Ning et al. 2020; Yan et al. 2021) or search frameworks (Luo et al. 2018; Ning et al. 2020; Shi et al. 2020).

Experiments and Results

In this section, we conduct experiments across different search spaces and under various experimental settings to evaluate the dynamic ensemble performance predictor.

Evaluation of Prediction Ability

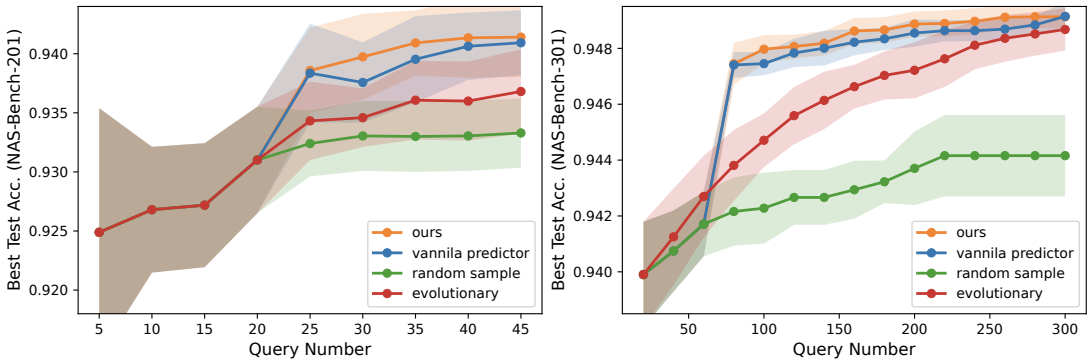
To begin with, we evaluate the prediction ability improvement brought by the proposed dynamic ensemble performance predictor on several public benchmarks with different training ratios and architecture encoders. These experiments mainly compare the ranking qualities of predictors.

Search Space. We conduct experiments on the five search spaces for a thorough evaluation: NAS-Bench-201, NAS-Bench-301, NDS-ResNet / ResNeXt-A and MobileNet-V3.

We divide architectures into a training and validation split for each space. We train the predictors on the former and test their prediction ability on the latter. All architectures in the training split with different types of low-fidelity information are used in the first training step. The detailed search space description, data split, types and acquisition of the utilized low-fidelity information are elaborated in the appendix.

Predictor Construction. In the basic predictor prediction flow, an encoder first encodes the architecture into an embedding vector. Then the vector is fed into an MLP to get the prediction score. We use LSTM (Luo et al. 2018) and GATES (Ning et al. 2020) as the encoder. We only use LSTM for the non-topological search spaces, including NDS-ResNet / ResNeXt-A and MobileNet-V3, since GATES is specially designed for topological architectures.

Training Settings. Following the previous studies (Ning et al. 2020; Xu et al. 2021), we train predictors with the hinge pair-wise ranking loss with margin $m = 0.1$. We first train different low-fidelity experts for 200 epochs and then finetune the dynamic ensemble performance predictor on the actual performance data for 200 epochs. For comparison, we directly train the vanilla predictor on the actual performance data for 200 epochs. An Adam optimizer with learning rate 1e-3 is applied for optimization. The batch sizes used for NAS-Bench-201, NAS-Bench-301, NDS and MobileNetV3 search spaces are 512, 128, 128 and 512, respectively.



(a) (b)

Figure 2: Comparison with other search strategies on NAS-Bench-201 (Figure 2a) and NAS-Bench-301 (Figure 2b). We report the test accuracy of the architecture with the highest reward among all sampled architectures.

Method	CIFAR-10 (%)		CIFAR-100 (%)		ImageNet-16-120 (%)	
	valid	test	valid	test	valid	test
RSPS (Li and Talwalkar 2019)	$84.16_{(1.69)}$	$87.66_{(1.69)}$	$59.00_{(4.60)}$	$58.33_{(4.34)}$	$31.56_{(3.28)}$	$31.14_{(3.88)}$
DARTS-V2 (Liu, Simonyan, and Yang 2018)	$39.77_{(0.00)}$	$54.30_{(0.00)}$	$15.03_{(0.00)}$	$15.61_{(0.00)}$	$16.43_{(0.00)}$	$16.32_{(0.00)}$
GDAS (Dong and Yang 2019b)	$90.00_{(0.21)}$	$93.51_{(0.13)}$	$71.15_{(0.27)}$	$70.61_{(0.26)}$	$41.70_{(1.26)}$	$41.84_{(0.90)}$
SETN (Dong and Yang 2019a)	$82.25_{(5.17)}$	$86.19_{(4.63)}$	$56.86_{(7.59)}$	$56.87_{(7.77)}$	$32.54_{(3.63)}$	$31.90_{(4.07)}$
ENAS-V2 (Pham et al. 2018)	$39.77_{(0.00)}$	$54.30_{(0.00)}$	$15.03_{(0.00)}$	$15.61_{(0.00)}$	$16.43_{(0.00)}$	$16.32_{(0.00)}$
Random Sample	$90.03_{(0.36)}$	$93.70_{(0.36)}$	$70.93_{(1.09)}$	$71.04_{(1.07)}$	$44.45_{(1.10)}$	$44.57_{(1.25)}$
NPENAS (Wei et al. 2020)	$91.08_{(0.11)}$	$91.52_{(0.16)}$	-	-	-	-
REA (Real et al. 2019)	$91.19_{(0.31)}$	$93.92_{(0.30)}$	$71.81_{(1.12)}$	$71.84_{(0.99)}$	$45.15_{(0.89)}$	$45.54_{(1.03)}$
NASBOT (White et al. 2020)	-	$93.64_{(0.23)}$	-	$71.38_{(0.82)}$	-	$45.88_{(0.37)}$
REINFORCE (Williams 1992)	$91.09_{(0.37)}$	$93.85_{(0.37)}$	$71.61_{(1.12)}$	$71.71_{(1.09)}$	$45.05_{(1.02)}$	$45.24_{(1.18)}$
BOHB (Falkner, Klein, and Hutter 2018)	$90.82_{(0.53)}$	$93.61_{(0.52)}$	$70.74_{(1.29)}$	$70.85_{(1.28)}$	$44.26_{(1.36)}$	$44.42_{(1.49)}$
ReNAS (Xu et al. 2021)	$90.90_{(0.31)}$	$93.99_{(0.25)}$	$71.96_{(0.99)}$	$72.12_{(0.79)}$	$45.85_{(0.47)}$	$45.97_{(0.49)}$
Ours	$\mathbf{91.59}_{(0.02)}$	$\mathbf{94.37}_{(0.00)}$	$\mathbf{73.49}_{(0.00)}$	$\mathbf{73.50}_{(0.00)}$	$\mathbf{46.41}_{(0.06)}$	$\mathbf{46.39}_{(0.01)}$
Optimal	91.61	94.37	73.49	73.51	46.77	47.31
ResNet	90.83	93.97	70.42	70.86	44.53	43.63

Table 3: Search results on NAS-Bench-201. The standard deviation is in the subscript.

Results. Following previous studies (Ning et al. 2021), we adopt Kendall's Tau (KD) as the evaluation criteria. As the results shown in Table 2, our proposed method outperforms *vanilla* predictor training consistently on different search spaces, architecture encoders, and training ratios. Especially, our method brings a larger improvement when the training ratio is smaller. For example, on NDS-ResNet and NDS-ResNeXt-A, our method achieves 0.7064 and 0.7753 Kendall's Tau with 1% training samples, respectively, much better than the vanilla predictor (0.2549, 0.3568).

Mitigating the Cold-Start Issue

We conduct architecture search on several search spaces to demonstrate that our method can effectively mitigate the cold-start issue and boost the performance of NAS.

Search on NAS-Bench-201. We conduct experiments on NAS-Bench-201 under three settings with architectures encoded by GATES. We use the validation accuracy on the CIFAR-10 dataset as the reward to guide the search.

Comparison with Different Search Strategies. We compare our method with random sample, tournament-based evolutionary ($\pi = 20, \mu = 5$), and predictor-based flow without utilizing low-fidelity information. Each method is run ten times. We set $N_0 = 20$, $M = 7813$, $T_p = 5$, $T_{pe} = 50$, $N_p = 5$, $\pi = 20$, $\mu = 5$ and $K = 100$ for our method. In total, we query the benchmark 45 times and report the test accuracy of the best architecture selected by the predictor. As the results shown in Figure 2a, the best architectures discovered by our method have higher test accuracies using the same query times.

Comparison with AceNAS and ProxyBO. We compare our method with ProxyBO (Shen et al. 2021) and AceNAS (Zhang et al. 2021) to verify that our method is a better way to utilize cheaper-to-obtain estimations. All search settings are kept the same as in other experiments, except for $N_p = 20$ and $T_p = 9$. In total, we query the benchmark 200 times and report the accuracy of the best architecture selected by the predictor. The search budget is the same as

Method	Test Error (%)			Param† (M)	FLOPs† (M)
	CIFAR-10	CIFAR-100	ImageNet (Top1/Top5)		
NASNet-A (Zoph et al. 2018)	2.65	17.81	26.0 / 8.4	3.3	564
PNAS (Liu et al. 2018)	3.41±0.09	17.63	25.8 / 8.1	3.2	588
EN^{2}AS (Zhang et al. 2020a)	2.61±0.06	16.45	26.7 / 8.9	3.1	506
NSAS (Zhang et al. 2020b)	2.59±0.06	17.56	25.5 / 8.2	3.1	506
DARTS (Liu, Simonyan, and Yang 2018)	2.76±0.09	17.54	26.9 / 8.7	3.4	574
GDAS (Dong and Yang 2019b)	2.93	18.38	26.0 / 8.5	3.4	545
SNAS (Xie et al. 2018)	2.85±0.02	20.09	27.3 / 9,2	2.8	474
PC-DARTS (Xu et al. 2019)	2.57±0.03	17.11	25.1 / 7.8	3.6	586
NAO (Luo et al. 2018)	2.48	15.67‡	25.7 / 8.2	10.6	584
GATES (Ning et al. 2020)	2.58	-	-	4.1	-
BANANAS (White, Neiswanger, and Savani 2019)	2.57	-	-	4.0	-
NPENAS-BO (Wei et al. 2020)	2.64 ± 0.08	-	-	-	-
NAS-BOWL (Ru et al. 2021)	2.61 ± 0.08	-	-	3.7	-
Ours	**2.30**	**16.07**	**24.4 / 7.4**	4.1	645

†: "Param" is the model size of CIFAR-10 model, while "FLOPs" is calculated based on the ImageNet models.
‡: This architecture is much larger than ours.

Table 4: Test error comparison with other NAS methods on CIFAR-10, CIFAR-100, and ImageNet.

Search Space	Vanilla	Ours	Random Sample	Optimal
NDS-ResNet	$0.9437_{(0.0000)}$	$\mathbf{0.9488}_{(0.0001)}$	$0.9420_{(0.0021)}$	0.9516
NDS-ResNeXt-A	$0.9454_{(0.0005)}$	$\mathbf{0.9456}_{(0.0000)}$	$0.9368_{(0.0038)}$	0.9483
MobileNet-V3	$0.7718_{(0.0000)}$	$\mathbf{0.7721}_{(0.0002)}$	$0.7633_{(0.0034)}$	0.7749

Table 5: Discovered Architecture accuracies. We report the average values and the standard deviation is in the subscript.

that of ProxyBO but less than AceNAS (500 queries). We run our method ten times with different seeds. Our method gets an 8.39% validation error on CIFAR-10 and 26.50% test error on CIFAR-100, respectively, better than ProxyBO (8.56%, 26.53%) and AceNAS (26.62% on CIFAR-100).

Comparison with Other NAS Methods. For a fair comparison with other NAS methods, following (Xu et al. 2021), we finetune the predictor in the second step with 90 randomly sampled architectures and their corresponding rewards. Then we traverse the search space with the predictor and report the best validation and test accuracies among the top-10 architectures selected by the predictor. We run the experiment 10 times and report the average and standard values[2]. As shown in Table 3, our method achieves better performance than the other methods on all three datasets. Remarkably, by utilizing one-shot estimation in predictor-based NAS, our method significantly outperforms the original one-shot method ENAS (Pham et al. 2018).

Search on NAS-Bench-301. With GATES as the encoder, we set $N_0 = 60$, $M = 5896$, $T_p = 10$, $T_{pe} = 100$, $N_p = 20$, $\pi = 20$, $\mu = 10$ and $K = 100$ for our method. In total, we query the benchmark 300 times. We compare our methods with random sample, tournament-based evolutionary ($\pi = 20$, $\mu = 10$), and the same predictor-based flow but without the utilization of low-fidelity information. Each method is run ten times with different seeds. The result is shown in Figure 2b. Our method achieves better test accuracies than predictor-based flow without the utilization of low-fidelity information and outperforms random sample and evolutionary methods by a large margin.

Search on DARTS. We further employ our method in the DARTS (Liu, Simonyan, and Yang 2018) search space. For fast experiments, we conduct architecture search on the NAS-Bench-301 benchmark, which is similar to the DARTS space. The search settings are the same as those on NAS-Bench-301. In total, we query the benchmark 300 times. After the search process, the best-discovered architecture is augmented following the DARTS setting and trained from scratch to get the final test accuracy. Detailed discovered architecture training settings are elaborated in the appendix. The comparison of the test errors is shown in Table 4. Our method achieves a test error of 2.30% on CIFAR-10, better than the previous one-shot NAS methods, such as DARTS (3.00%) and GDAS (2.93%). When transferred to CIFAR-100 and ImageNet, the discovered architecture achieves a test error of 16.07% and 24.4%, respectively, also outperforming the other architectures.

Search on MobileNet-V3 and NDS. After the first training step, we finetune the entire predictor on 1% architectures in the training split. On MobileNet-V3, we traverse 100000 randomly sampled architectures with the predictor.

[2]Following ReNAS (Xu et al. 2021), we report the predictor training time as the search cost.

Search Space	Manner	Proportions of training samples		
		1%	5%	10%
NAS-Bench-201	Simple	$0.6936_{(0.0038)}$	$0.7763_{(0.0058)}$	$0.8218_{(0.0015)}$
	Uniform	$0.7442_{(0.0031)}$	$0.8296_{(0.0019)}$	$0.8549_{(0.0004)}$
	Ours	$\mathbf{0.7835}_{(0.0062)}$	$\mathbf{0.8538}_{(0.0029)}$	$\mathbf{0.8683}_{(0.0015)}$
NDS ResNet	Simple	$0.5789_{(0.0145)}$	$0.7247_{(0.0088)}$	$0.7349_{(0.0125)}$
	Uniform	$0.6794_{(0.0174)}$	$0.7302_{(0.0055)}$	$0.7452_{(0.0052)}$
	Ours	$\mathbf{0.7064}_{(0.0109)}$	$\mathbf{0.7548}_{(0.0080)}$	$\mathbf{0.7652}_{(0.0037)}$
NDS ResNeXt-A	Simple	$0.7326_{(0.0122)}$	$0.7942_{(0.0073)}$	$0.8009_{(0.0042)}$
	Uniform	$0.7694_{(0.0062)}$	$0.8253_{(0.0033)}$	$0.8348_{(0.0040)}$
	Ours	$\mathbf{0.7753}_{(0.0010)}$	$\mathbf{0.8276}_{(0.0024)}$	$\mathbf{0.8398}_{(0.0044)}$

Table 6: The Kendall's Tau (average over five runs) of using the LSTM encoder on NAS-Bench-201 and NDS-ResNet / ResNeXt-A. The standard deviation is in the subscript. "Uniform" represents learning a uniform set of weight coefficients for all the architectures. "Simple" represents simply averaging outputs of different low-fidelity experts.

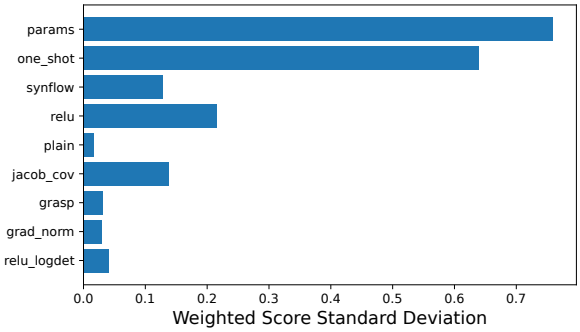


Figure 3: Standard deviation of weighted scores of different low-fidelity experts on NDS-ResNet.

On NDS-ResNet / ResNeXt-A, we traverse all the architectures in the search space. The best test accuracy among the top-10 architectures selected by the predictor is reported. As shown in Table 5, compared with other strategies, our method consistently discovers superior architectures.

Efficiency Comparison with Other Methods. Except for the one-shot score, most types of low-fidelity information can be obtained at an extremely low cost. For example, evaluation of the parameter size for all architectures in NAS-Bench-201 can be accomplished within a minute. Although utilizing the one-shot information involves supernet training and submodel testing, our method is still more efficient than baselines. For example, the cost of training the supernet and testing 7813 candidate architectures on NAS-Bench-201 is comparable to training about 15 architectures for 200 epochs. In our experiment, when querying 45 architectures to get their ground-truth performance in NAS-Bench-201, the equivalent total cost is about training 45+15=60 architectures. And the accuracy of our discovered architecture (94.09%) surpasses the architecture accuracy (93.99%) discovered by ReNAS after 90 queries by a large margin.

Empirical Analysis

Dynamic Ensemble Analysis. Since we model predictor learning as a ranking problem, the absolute value of the weighted score by an expert does not reflect its importance directly. This is because the output range of experts varies. If there is a low-fidelity expert that has the highest weighting coefficient, whose weighted scores for all architectures are the same. Then, this expert does not contribute new information to the architecture ranking. In other words, only the difference between weighted scores of architectures by an expert contributes to the architecture ranking. So, instead, we calculate the standard deviation of the weighted scores k_i as the criterion. Figure 3 shows the results of different low-fidelity experts on NDS-ResNet. The experts on the parameter size and one-shot score have a much larger standard deviation than other types, indicating that the predictor highly relies on them for prediction. On the other hand, in the appendix, we empirically verify that these two low-fidelity information types are the most beneficial ones on NDS-ResNet. That is to say, the relative importance of low-fidelity experts in our predictor aligns well with the extent of benefits brought by low-fidelity information when only one type of information is used. This backs the rationality of using our method to automatically and adequately combine different low-fidelity information.

Comparison with Uniform Weight Learning. An alternative to the dynamic ensemble is to learn a uniform set of coefficients for all the architecture in the search space. However, considering different types of low-fidelity information have different prediction abilities for different regions of the search space, the weighting coefficients for architectures would better be different. To verify this intuition, we conduct experiments on NAS-Bench-201 and NDS ResNet / ResNeXt-A with the LSTM encoder for comparison. In addition, we also make comparison with a simple ensemble method that just averages outputs of different low-fidelity experts. As shown in Table 6, the dynamic ensemble method consistently outperforms the uniform and simple ensemble method, demonstrating the effectiveness of our method.

Conclusion

This paper proposes to leverage low-fidelity information to mitigate the "cold-start" problem of predictor-based NAS. Despite the intuitiveness of this idea, we observe that utilizing inappropriate low-fidelity information might damage the prediction ability and different search spaces have different preferences for the utilized low-fidelity information type. To circumvent the need to manually decide on which low-fidelity information to use for each architecture and search space, we propose a dynamic ensemble predictor framework to fuse beneficial information from different low-fidelity experts automatically. Experiments across five search spaces with different architecture encoders under various experimental settings demonstrate the effectiveness of our methods. Our method can be easily incorporated with existing predictor-based NAS methods to boost search performances.

Acknowledgments

This work was supported by National Natural Science Foundation of China (No. U19B2019, 61832007), Huawei Noah's Ark, Beijing National Research Center for Information Science and Technology (BNRist), Tsinghua EE Xilinx AI Research Fund, and Beijing Innovation Center for Future Chips. We thank the anonymous reviewers for their constructive suggestions.

References

Abdelfattah, M. S.; Mehrotra, A.; Dudziak, Ł.; and Lane, N. D. 2021a. Zero-Cost Proxies for Lightweight NAS. In *International Conference on Learning Representations*.

Abdelfattah, M. S.; Mehrotra, A.; Dudziak, Ł.; and Lane, N. D. 2021b. Zero-cost proxies for lightweight NAS. *arXiv preprint arXiv:2101.08134*.

Bender, G.; Kindermans, P.-J.; Zoph, B.; Vasudevan, V.; and Le, Q. 2018. Understanding and simplifying one-shot architecture search. In *International Conference on Machine Learning*, 550–559. PMLR.

Cai, H.; Gan, C.; Wang, T.; Zhang, Z.; and Han, S. 2019. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*.

Chen, Y.; Yang, T.; Zhang, X.; Meng, G.; Xiao, X.; and Sun, J. 2019. DetNAS: Backbone search for object detection. In *Advances in Neural Information Processing Systems*, 6638–6648.

Dong, X.; and Yang, Y. 2019a. One-Shot Neural Architecture Search via Self-Evaluated Template Network. *arXiv preprint arXiv:1910.05733*.

Dong, X.; and Yang, Y. 2019b. Searching for A Robust Neural Architecture in Four GPU Hours. *arXiv preprint arXiv:1910.04465*.

Dong, X.; and Yang, Y. 2020. NAS-Bench-201: Extending the Scope of Reproducible Neural Architecture Search. In *International Conference on Learning Representations*.

Dudziak, Ł.; Chau, T.; Abdelfattah, M. S.; Lee, R.; Kim, H.; and Lane, N. D. 2020. Brp-nas: Prediction-based nas using gcns. *arXiv preprint arXiv:2007.08668*.

Elsken, T.; Metzen, J. H.; and Hutter, F. 2019. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1): 1997–2017.

Falkner, S.; Klein, A.; and Hutter, F. 2018. BOHB: Robust and Efficient Hyperparameter Optimization at Scale. *arXiv preprint arXiv:1807.01774*.

Guo, Z.; Zhang, X.; Mu, H.; Heng, W.; Liu, Z.; Wei, Y.; and Sun, J. 2020. Single path one-shot neural architecture search with uniform sampling. In *European Conference on Computer Vision*, 544–560. Springer.

Huawei. 2020. Mindspore. https://www.mindspore.cn/. Accessed: 2023-04-02.

Kandasamy, K.; Neiswanger, W.; Schneider, J.; Póczos, B.; and Xing, E. P. 2018. Neural architecture search with Bayesian optimisation and optimal transport. In *Advances in neural information processing systems*, 2020–2029.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images. *Handbook of Systemic Autoimmune Diseases*, 1(4).

Lee, N.; Ajanthan, T.; and Torr, P. H. S. 2018. SNIP: Single-shot Network Pruning based on Connection Sensitivity. *arXiv preprint arXiv:1810.02340*.

Li, L.; and Talwalkar, A. 2019. Random Search and Reproducibility for Neural Architecture Search. *arXiv preprint arXiv:1902.07638*.

Lin, M.; Wang, P.; Sun, Z.; Chen, H.; Sun, X.; Qian, Q.; Li, H.; and Jin, R. 2021. Zen-NAS: A Zero-Shot NAS for High-Performance Deep Image Recognition. In *IEEE International Conference on Computer Vision*, 347–356.

Liu, C.; Zoph, B.; Neumann, M.; Shlens, J.; Hua, W.; Li, L.-J.; Fei-Fei, L.; Yuille, A.; Huang, J.; and Murphy, K. 2018. Progressive neural architecture search. In *European Conference on Computer Vision*, 19–34.

Liu, H.; Simonyan, K.; and Yang, Y. 2018. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*.

Luo, R.; Tian, F.; Qin, T.; Chen, E.; and Liu, T.-Y. 2018. Neural architecture optimization. In *Advances in neural information processing systems*, 7827–7838.

Mellor, J.; Turner, J.; Storkey, A.; and Crowley, E. J. 2020. Neural Architecture Search without Training. *arXiv preprint arXiv:2006.04647*.

Mozer, M.; and Smolensky, P. 1988. Skeletonization: a technique for trimming the fat from a network via relevance assessment. In *Advances in Neural Information Processing Systems*.

Ning, X.; Tang, C.; Li, W.; Zhou, Z.; Liang, S.; Yang, H.; and Wang, Y. 2021. Evaluating Efficient Performance Estimators of Neural Architectures. In *Advances in Neural Information Processing Systems*.

Ning, X.; Zheng, Y.; Zhao, T.; Wang, Y.; and Yang, H. 2020. A generic graph-based neural architecture encoding scheme for predictor-based nas. In *European Conference on Computer Vision*, 189–204. Springer.

Ning, X.; Zhou, Z.; Zhao, J.; Zhao, T.; Deng, Y.; Tang, C.; Liang, S.; Yang, H.; and Wang, Y. 2022. TA-GATES: An

Encoding Scheme for Neural Network Architectures. In *Advances in Neural Information Processing Systems*.

Pham, H.; Guan, M.; Zoph, B.; Le, Q.; and Dean, J. 2018. Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning*, 4095–4104. PMLR.

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

Radosavovic, I.; Johnson, J.; Xie, S.; Lo, W.-Y.; and Dollár, P. 2019. On network design spaces for visual recognition. In *IEEE International Conference on Computer Vision*, 1882–1890.

Real, E.; Aggarwal, A.; Huang, Y.; and Le, Q. V. 2019. Regularized evolution for image classifier architecture search. In *AAAI Conference on Artificial Intelligence*, volume 33, 4780–4789.

Ru, B.; Wan, X.; Dong, X.; and Osborne, M. 2021. Interpretable Neural Architecture Search via Bayesian Optimisation with Weisfeiler-Lehman Kernels. In *International Conference on Learning Representations*.

Shen, Y.; Li, Y.; Zheng, J.; Zhang, W.; Yao, P.; Li, J.; Yang, S.; Liu, J.; and Cui, B. 2021. ProxyBO: Accelerating Neural Architecture Search via Bayesian Optimization with Zero-cost Proxies. *arXiv preprint arXiv:2110.10423*.

Shi, H.; Pi, R.; Xu, H.; Li, Z.; Kwok, J.; and Zhang, T. 2020. Bridging the gap between sample-based and one-shot neural architecture search with bonas. *Advances in Neural Information Processing Systems*, 33.

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

Tanaka, H.; Kunin, D.; Yamins, D. L.; and Ganguli, S. 2020. Pruning neural networks without any data by iteratively conserving synaptic flow. *arXiv preprint arXiv:2006.05467*.

Tang, Y.; Wang, Y.; Xu, Y.; Chen, H.; Shi, B.; Xu, C.; Xu, C.; Tian, Q.; and Xu, C. 2020. A semi-supervised assessor of neural architectures. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1810–1819.

Theis, L.; Korshunova, I.; Tejani, A.; and Huszár, F. 2018. Faster gaze prediction with dense networks and Fisher pruning. *arXiv preprint arXiv:1801.05787*.

Wang, C.; Zhang, G.; and Grosse, R. 2020. Picking Winning Tickets Before Training by Preserving Gradient Flow. *arXiv preprint arXiv:2002.07376*.

Wang, N.; Gao, Y.; Chen, H.; Wang, P.; Tian, Z.; Shen, C.; and Zhang, Y. 2020. NAS-FCOS: Fast neural architecture search for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 11943–11951.

Wei, C.; Niu, C.; Tang, Y.; Wang, Y.; Hu, H.; and Liang, J. 2020. Npenas: Neural predictor guided evolution for neural architecture search. *arXiv preprint arXiv:2003.12857*.

White, C.; Neiswanger, W.; Nolen, S.; and Savani, Y. 2020. A Study on Encodings for Neural Architecture Search. *arXiv preprint arXiv:2007.04965*.

White, C.; Neiswanger, W.; and Savani, Y. 2019. Bananas: Bayesian optimization with neural architectures for neural architecture search. *arXiv preprint arXiv:1910.11858*, 1(2).

White, C.; Zela, A.; Ru, R.; Liu, Y.; and Hutter, F. 2021. How powerful are performance predictors in neural architecture search? *Advances in Neural Information Processing Systems*, 34: 28454–28469.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4): 229–256.

Wu, B.; Dai, X.; Zhang, P.; Wang, Y.; Sun, F.; Wu, Y.; Tian, Y.; Vajda, P.; Jia, Y.; and Keutzer, K. 2019. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *IEEE Conference on Computer Vision and Pattern Recognition*, 10734–10742.

Wu, J.; Dai, X.; Chen, D.; Chen, Y.; Liu, M.; Yu, Y.; Wang, Z.; Liu, Z.; Chen, M.; and Yuan, L. 2021. Stronger nas with weaker predictors. *Advances in Neural Information Processing Systems*, 34: 28904–28918.

Xie, S.; Zheng, H.; Liu, C.; and Lin, L. 2018. SNAS: Stochastic Neural Architecture Search. *arXiv preprint arXiv:1812.09926*.

Xu, Y.; Wang, Y.; Han, K.; Tang, Y.; Jui, S.; Xu, C.; and Xu, C. 2021. ReNAS: Relativistic evaluation of neural architecture search. In *IEEE Conference on Computer Vision and Pattern Recognition*, 4411–4420.

Xu, Y.; Xie, L.; Zhang, X.; Chen, X.; Qi, G.-J.; Tian, Q.; and Xiong, H. 2019. PC-DARTS: Partial channel connections for memory-efficient architecture search. *arXiv preprint arXiv:1907.05737*.

Yan, S.; Song, K.; Liu, F.; and Zhang, M. 2021. Cate: Computation-aware neural architecture encoding with transformers. In *International Conference on Machine Learning*, 11670–11681. PMLR.

Yan, S.; Zheng, Y.; Ao, W.; Zeng, X.; and Zhang, M. 2020. Does unsupervised architecture representation learning help neural architecture search? *Advances in Neural Information Processing Systems*, 33: 12486–12498.

Zhang, M.; Jiang, S.; Cui, Z.; Garnett, R.; and Chen, Y. 2019. D-vae: A variational autoencoder for directed acyclic graphs. *Advances in Neural Information Processing Systems*, 32.

Zhang, M.; Li, H.; Pan, S.; Chang, X.; Ge, Z.; and Su, S. W. 2020a. Differentiable Neural Architecture Search in Equivalent Space with Exploration Enhancement. In *Advances in Neural Information Processing Systems*.

Zhang, M.; Li, H.; Pan, S.; Chang, X.; and Su, S. 2020b. Overcoming multi-model forgetting in one-shot NAS with diversity maximization. In *IEEE Conference on Computer Vision and Pattern Recognition*, 7809–7818.

Zhang, Y.; Yan, C.; Zhang, Q.; Zhang, L. L.; Yang, Y.; Gao, X.; and Yang, Y. 2021. Acenas: Learning to rank ace neural architectures with weak supervision of weight sharing. *arXiv preprint arXiv:2108.03001*.

Zoph, B.; and Le, Q. V. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.

Zoph, B.; Vasudevan, V.; Shlens, J.; and Le, Q. V. 2018. Learning transferable architectures for scalable image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 8697–8710.