ProxyBO: Accelerating Neural Architecture Search via Bayesian Optimization with Zero-Cost Proxies

Yu Shen[1,4], **Yang Li**[5], **Jian Zheng**[3], **Wentao Zhang**[6,7], **Peng Yao**[4], **Jixiang Li**[4], **Sen Yang**[4], **Ji Liu**[4], **Bin Cui**[1,2]

[1]Key Lab of High Confidence Software Technologies, Peking University, China
[2]Institute of Computational Social Science, Peking University (Qingdao), China
[3]School of Computer Science and Engineering, Beihang University, China
[4]Kuaishou Technology, China
[5]Data Platform, TEG, Tencent Inc., China
[6]Mila - Québec AI Institute
[7]HEC, Montréal, Canada
{shenyu,bin.cui}@pku.edu.cn, zhengjian2322@buaa.edu.cn,
{yaopeng,lijixiang,senyang}@kuaishou.com, jiliu@kwai.com,
thomasyngli@tencent.com, wentao.zhang@mila.quebec

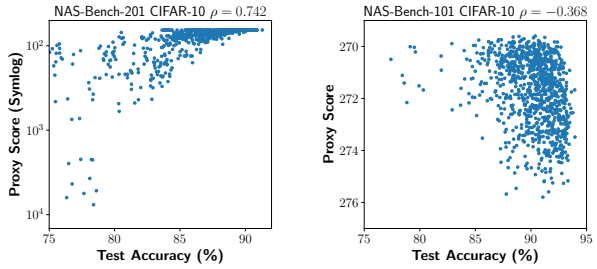
Abstract

Designing neural architectures requires immense manual efforts. This has promoted the development of neural architecture search (NAS) to automate the design. While previous NAS methods achieve promising results but run slowly, zero-cost proxies run extremely fast but are less promising. Therefore, it's of great potential to accelerate NAS via those zero-cost proxies. The existing method has two limitations, which are *unforeseeable reliability* and *one-shot usage*. To address the limitations, we present ProxyBO, an efficient Bayesian optimization (BO) framework that utilizes the zero-cost proxies to accelerate neural architecture search. We apply the generalization ability measurement to estimate the fitness of proxies on the task during each iteration and design a novel acquisition function to combine BO with zero-cost proxies based on their dynamic influence. Extensive empirical studies show that ProxyBO consistently outperforms competitive baselines on five tasks from three public benchmarks. Concretely, ProxyBO achieves up to $5.41\times$ and $3.86\times$ speedups over the state-of-the-art approaches REA and BRP-NAS.

Introduction

Discovering state-of-the-art neural architectures (He et al. 2016; Huang et al. 2017) requires substantial efforts of human experts. The manual design is often costly and becomes increasingly expensive when networks grow larger. Recently, the neural network community has witnessed the development of neural architecture search (NAS) (Zoph et al. 2018; Real et al. 2019; Liu, Simonyan, and Yang 2019; Cai et al. 2018), which turns the design of architectures into an optimization problem without human interaction and achieves promising results in a wide range of fields, such as image classification (Zoph et al. 2018; Real et al. 2019), sequence modeling (Pham et al. 2018; So, Le, and Liang 2019), etc.

Bayesian optimization (BO) (Hutter, Hoos, and Leyton-Brown 2011; Snoek, Larochelle, and Adams 2012) has emerged as a state-of-the-art method for NAS (Ying et al. 2019; White, Neiswanger, and Savani 2021). It trains a predictor, namely surrogate, on observations and selects the next architecture to evaluate based on its predictions. Recent work differs in the choice of surrogate, including Bayesian neural networks (Springenberg et al. 2016), Graph neural networks (Ma, Cui, and Yang 2019), etc. Despite the promising converged results, they share the common drawback that training a well-performed surrogate requires a sufficient number of evaluations, which often take days to obtain.



(a) NAS-Bench-201 CIFAR-10 (b) NAS-Bench-101

Figure 1: Spearman ρ of `jacob_cov` over NAS search spaces using 1000 randomly sampled architectures.

While several approaches attempt to reduce the evaluation cost via weight sharing (Pham et al. 2018) or gradient decent (Liu, Simonyan, and Yang 2019), recent work (Chen, Gong, and Wang 2021; Mellor et al. 2021) proposes several zero-cost proxies to estimate the performance of architecture at initialization using only a few seconds instead of network training. Though they achieve less promising results than BO-based methods, the computation of zero-cost proxies is speedy. Then, there comes up a question: "*Can we speed up NAS by combining the advantages of both Bayesian optimization and zero-cost proxies, i.e., achieving promising results with fewer computationally expensive evaluations?*"

Opportunities. As shown in Figure 1(a), the Spearman cor-

Copyright © 2023, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

relation coefficient between the proxy `jacob_cov` and test accuracy on NAS-Bench-201 CIFAR-10 is 0.742. Since the coefficient is a relatively large positive value, the proxy can be applied to rank architectures and guide the selection of the next architecture to evaluate.

Challenges. First, utilizing zero-cost proxies is non-trivial. Rather than applying Bayesian optimization, recent work (Abdelfattah et al. 2021) attempts to perform a simple warm-up on binary relation predictors based on a specific proxy. However, the proxies are not fully utilized via warm-up and may even lead to negative effects due to two limitations: ***L1. Unforeseeable reliability***. The warm-up method chooses to apply the best proxy found by exhaustively evaluating thousands of architectures over a specific search space. However, in practice, the correlation between proxy scores and objective values is unknown before searching. As shown in Figure 1, although the proxy `jacob_cov` works well on NAS-Bench-201, it performs worse than ranking randomly on NAS-Bench-101. In this case, using this proxy may lead to negative effects; ***L2. One-shot usage***. The warm-up method applies the proxies only once before searching by pre-training the neural binary predictor based on proxy scores. As a result, the influence of proxies in the warm-up method is irreversible, making it difficult to identify and get rid of those potentially bad proxies during searching. Due to the limitations, how to unleash the potential of the proxies is still an open question.

In addition, though BO and zero-cost proxies may complement with each other, combining the two parts is also non-trivial. As a learning model, the BO surrogate generalizes better with more evaluation results while the ranking ability of proxies is constant, which is only determined by the current task. In other words, zero-cost proxies bring benefits when few evaluations are given, but they become less helpful when the BO surrogate becomes accurate with sufficient evaluations. Therefore, the influence of the proxies should be decreased during optimization, and a dynamic design should be made to match this trend.

Contributions. In this paper, we propose ProxyBO, an efficient Bayesian optimization (BO) framework that utilizes the zero-cost proxies to significantly accelerate neural architecture search. The contributions are summarized as follows: 1) To the best of our knowledge, ProxyBO is the first method that utilizes the zero-cost proxies without prior knowledge about whether the proxies are suitable for the current task; 2) To deal with **L1**, we propose to estimate the reliability of different proxies by measuring their fitness during optimization. For **L2**, we apply a novel acquisition function to combine BO with the proxies based on their dynamic influence; 3) Empirical results on four public benchmarks showcase the superiority of ProxyBO compared with **fifteen** competitive baselines. Concretely, ProxyBO achieves up to $5.41\times$ and $3.86\times$ speedups over the state-of-the-art approaches REA and BRP-NAS, respectively.

Related Work

Designing neural architectures manually is often a challenging and time-consuming task since it is quite difficult for human experts to choose the proper operations and place each connection. As a result, this arouses great interest from both academia and industry to design neural architectures in an automatic manner. Inspired by successful hand-crafted architectures, pioneering work (Zoph et al. 2018) manually designs a fixed macro-structure, which is composed of stacks of cells (micro-structure). The search is then conducted over those cells instead of the whole architecture via different search strategies, including reinforcement learning (Zoph et al. 2018; Pham et al. 2018), evolutionary algorithm (Real et al. 2019), gradient decent (Liu, Simonyan, and Yang 2019), binary relation predictor (Dudziak et al. 2020), etc.

Among various approaches proposed for neural architecture search, recent researches (White, Neiswanger, and Savani 2021; Ying et al. 2019; Siems et al. 2020) have shown the competitive performance of Bayesian optimization (BO) with a performance predictor. The original BO (Snoek, Larochelle, and Adams 2012) is proposed for solving black-box optimization, in which the output can only be obtained by an objective function, and no extra information like derivatives is available. As evaluating the validation performance of a given neural architecture is also a black-box process, BO can be directly applied to search for neural architectures. Benchmark studies (Ying et al. 2019; Siems et al. 2020) point out that SMAC (Hutter, Hoos, and Leyton-Brown 2011), a classical BO approach, achieves state-of-the-art performance given enough budgets. BANANAS (White, Neiswanger, and Savani 2021) digs deeper into the BO framework and compares each part of the framework via extensive experiments. Other work further improves BO by combining the characteristics of neural architectures, e.g., NASBOT (Kandasamy et al. 2018) defines a pseudo-distance for kernel functions while GPWL (Ru et al. 2021) adopts the Weisfeiler-Lehman kernel. However, these methods share the same drawback that a sufficient number of evaluations are required to guide the BO framework. As the evaluation of architectures is time-consuming, the cost of the search algorithm is exorbitant.

To reduce the costs of NAS, several techniques have been applied in the literature. ENAS (Pham et al. 2018) applies the weight sharing strategy by sharing weights in the same operation. DARTS (Liu, Simonyan, and Yang 2019) models NAS as training an over-parameterized architecture including all candidate paths. EcoNAS (Zhou et al. 2020) investigates proxies with reduced resources during evaluation, e.g., fewer epochs, training samples, etc. BOHB (Falkner, Klein, and Hutter 2018) combines Hyperband (Li et al. 2017) and BO by early-stopping bad evaluations while MFES (Li et al. 2021a, 2022a) improves BOHB by taking all evaluations of different resource levels into consideration when sampling new configurations to evaluate. Transfer learning (Lee, Hyung, and Hwang 2021) are applied to learn from previous tasks.

Recent studies further accelerate NAS by estimating the performance of architectures at initialization (i.e., zero-cost proxies (Xu et al. 2021; Shu et al. 2021)). Synaptic saliency metrics (Lee, Ajanthan, and Torr 2018; Wang, Zhang, and Grosse 2020; Tanaka et al. 2020) measures the loss change when removing a certain parameter. Fisher (Theis et al. 2018) estimates the loss change when removing activation channels. NASWOT (Mellor et al. 2021) and TE-NAS (Chen, Gong, and Wang 2021) model the expressivity of architectures based on activations. HNAS (Shu et al. 2022) boosts

training-free NAS in a principled way. As proxy scores are somehow related to the ground-truth performance, recent work (Abdelfattah et al. 2021) proposes to warm up a neural binary relation predictor or select candidates for evolutionary algorithms using a specific proxy and achieves acceleration on benchmarks. OMNI (White et al. 2021) combines proxies into the BO surrogate. Due to the limitations (**L1** and **L2**), how to unleash the potential of zero-cost proxies without prior knowledge is still an open question.

Preliminary

As stated above, neural architecture search (NAS) can be modeled as a black-box optimization problem. The goal is to solve $argmin_{x \in \mathcal{X}} f_{obj}(x)$ over an architecture search space $\mathcal{X}$, where $f_{obj}(x)$ is the objective performance metric (e.g., classification error on the validation set) corresponding to the architecture configuration x. In the following, we first introduce the framework of Bayesian optimization and then the zero-cost proxies used in our proposed framework.

Bayesian Optimization

To solve black-box optimization problems with expensive evaluation costs, Bayesian optimization (BO) follows the framework of sequential model-based optimization. A typical BO iteration loops over the following three steps: 1) BO fits a probabilistic surrogate model M based on the observations $D = \{(x_1, y_1), ..., (x_{n-1}, y_{n-1})\}$, in which x_i is the configuration evaluated in the i-th iteration and y_i is its corresponding observed performance; 2) BO uses the surrogate M to select the most promising configuration x_n by maximizing $x_n = \arg\max_{x \in \mathcal{X}} a(x; M)$, where $a(x; M)$ is the acquisition function designed to balance the trade-off between exploration and exploitation; 3) BO evaluates the configuration x_n to obtain y_n(i.e., train the architecture and obtain its validation performance), and augment the observations $D = D \cup \{(x_n, y_n)\}$.

We adopt the Probabilistic Random Forest (Hutter, Hoos, and Leyton-Brown 2011) as the surrogate model and the Expected Improvement (EI) (Jones, Schonlau, and Welch 1998) as the acquisition function. The EI is defined as:

$$a(x; M) = \int_{-\infty}^{\infty} \max(y_{best} - y, 0) p_M(y|x) dy, \qquad (1)$$

where $p_M(y|x)$ is the conditional probability of y given x under the surrogate model M, and y_{best} is the best observed performance in observations D, i.e., $y_{best} = \min\{y_1, ..., y_n\}$. Note that, EI only takes the improvement over the best performance into consideration.

Zero-cost Proxy

Different from the low-cost proxies (Zhou et al. 2020) which require less training resources, zero-cost proxies are a type of proxies that can be computed at initialization. The initial design goal of zero-cost proxies is to better direct exploration in existing NAS algorithms without the expensive training costs. Recent researches (Abdelfattah et al. 2021; Krishnakumar et al. 2022) show that a certain combination of proxies work better than single ones but proxies perform quite differently across different tasks. To ensure safe use of proxies, we consider multiple proxies so that at least one is likely to be helpful. In the following, we describe three metrics with their properties used in our proposed method.

`snip` (Lee, Ajanthan, and Torr 2018) is a saliency metric that approximates the loss change when a certain parameter is removed. `synflow` (Tanaka et al. 2020) optimizes `snip` to avoid layer collapse when performing parameter pruning. While `snip` that requires a batch of data and the original loss function, `synflow` computes the product of all parameters as its loss function and thus requires no data. The formulations are as follows,

$$\texttt{snip}: S(\theta) = \left| \frac{\partial L_{snip}}{\partial \theta} \odot \theta \right|, \ \texttt{synflow}: S(\theta) = \frac{\partial L_{syn}}{\partial \theta} \odot \theta, \qquad (2)$$

where L_{snip} is the loss function of a network, L_{syn} is the product of all parameters, and $\odot$ is the Hadamard product. While both `snip` and `synflow` are per-parameter metrics, we extend them to score the entire architecture x following (Abdelfattah et al. 2021) as $P(x) = -\sum_{\theta \in \Theta} S(\theta)$, where Θ refers to all the parameters of architecture x.

Recent work (Mellor et al. 2021) introduces a correlation metric `jacob_cov` that captures the correlation of activations of different inputs within a network given a batch of data. We refer to the original paper for the detailed derivation of the metric. The lower the correlation is, the better the network is expected to be.

The three proxies are selected due to two considerations: 1) All of the three proxies can be computed in a relatively short time using at most a batch of data; 2) The proxies have their own properties, i.e., `jacob_cov` highlights the activations while `snip` and `synflow` are gradient-based proxies of different inputs.

The Proposed Method

In this section, we present ProxyBO – our proposed method for efficient Bayesian optimization (BO) with zero-cost proxies. In general, ProxyBO alters the sampling procedure of traditional BO to integrate the information from zero-cost proxies, thus generalizing to different underlying BO algorithms. To tackle the challenges in the introduction, we will answer the following two questions: 1) how to measure the generalization ability of zero-cost proxies as well as the BO surrogate without prior knowledge, and 2) how to effectively integrate BO with zero-cost proxies during optimization.

Generalization Ability Measurement

The goal of Bayesian optimization (BO) is to iteratively find the best configuration with the optimal objective value. In other words, a proxy or a surrogate is helpful if it can order the performance of the given architecture configurations correctly. Inspired by the concept of some multi-fidelity techniques to measure the fitness of probabilistic surrogates (Li et al. 2021a), in our framework, we apply a measurement to dynamically estimate the usefulness of zero-cost proxies and BO surrogates during optimization. For simplicity, in the following discussion, we assume that the given objective function needs to be minimized. And the smaller the proxy score is, the better the architecture is expected to be.

For zero-cost proxies, we need to measure their ability to fit the ground-truth observations D, and thus we apply the Kendall-tau correlation to measure the correctness of their ranking results. To convert the value range to $[0, 1]$, we define the metric G as $G(P_i; D) = \frac{\tau_{P_i}+1}{2}$. τ_{P_i} is the coefficient of the pair set $\{(P_i(x), y)|(x, y) \in D\}$, where $P_i(x)$ is the output of the zero-cost proxy i given the architecture configuration x, and y is its corresponding ground-truth performance.

For BO surrogate, since it is directly trained on the observations D, the above definition only calculates the in-sample error of the BO surrogate and cannot correctly reflect the generalization ability of the surrogate on unseen data points. Therefore, we apply the k-fold cross-validation strategy for the BO surrogate. Denote f as the mapping function that maps an observed configuration x to its corresponding fold index as $f(x)$. The measurement for the BO surrogate is calculated as, $G(M; D) = \frac{\tau_M+1}{2}$. τ_M is the coefficient of the pair set $\{(M_{-f(x)}(x), y)|(x, y) \in D\}$, where $M_{-f(x)}$ refers to the predictive mean of the surrogate trained on observations D with the $f(x)$-th fold left out. In this way, x is not used when generating $M_{-f(x)}$, thus the new definition is able to measure the generalization ability of BO surrogate only by using the observations D. Through this measurement, ProxyBO is able to judge the surrogate during optimization.

Note that, to measure the generalization ability, the Kendall-tau is more appropriate than other intuitive alternatives such as mean squared error or log-likelihood because we do not care about the actual values of the predictions during the optimization. Instead, the framework only needs to identify the location of the optimum.

Dynamic Influence Combination

The original BO selects the next configuration to evaluate by maximizing its acquisition function. However, the BO surrogate is under-fitted when given few observations, i.e., it can not precisely predict the performance of unseen configurations to guide the selection of configurations. Rather than a machine learning model, the zero-cost proxies are formulated metrics, among which some may perform better than the under-fitted surrogate at the beginning of the optimization. On the other hand, as the number of observations grows over time, the generalization ability of BO surrogate gradually outperforms the proxies. In this case, more attention should be paid to the surrogate when selecting the next configuration to evaluate. Therefore, the dynamic influence of the proxies and the surrogate on configuration selection should be considered, and the design is non-trivial.

Concretely in ProxyBO, we alter the acquisition function for selecting configurations by combining the influence of each component. Though there are various methods to combine probabilistic outputs of similar scales (e.g., gPOE (Cao and Fleet 2014)), they can not be directly applied in ProxyBO, as we find that the outputs of proxies are deterministic, and they are of significantly different scales with the surrogate output. As a result, we propose to use **the sum of ranking** instead of directly adding the outputs. During each BO iteration, we sample Q configurations by random sampling and local sampling on well-performed observed configurations.

Algorithm 1: Pseudo code for *Sample* in ProxyBO

Input: the observations D, the current number of iteration T, the number of sampled configurations Q, the Bayesian optimization surrogate M, the zero-cost proxies $P_{1:K}$, and the temperature hyper-parameter τ_0.
Output: the next architecture configuration to evaluate.
1: **if** $|D| < 5$, then **return** a random configuration.
2: compute $G(\cdot; D)$ for each proxy and the surrogate.
3: compute $I(\cdot; D)$ according to Eq. 4.
4: draw Q configurations via random and local sampling.
5: compute EI based on surrogate M according to Eq. 1, and the proxy values for $P_{1:K}$ for each sampled configuration.
6: rank the Q configurations and obtain the ranking value of configuration x_j as $R_M(x_j)$ and $R_{P_i}(x_j)$ for the i-th proxy.
7: calculate the combined ranking $CR(x_j)$ for each configuration x_j according to Eq. 3.
8: **return** the configuration with the lowest CR value.

Then we calculate the EI and proxy values for each sampled configuration. Based on these values, we further rank the Q configurations and obtain the ranking value of x_j as $R_M(x_j)$ and $R_{P_i}(x_j)$ for the BO surrogate and proxies, respectively. Finally, the combined ranking value of x_j is defined as:

$$CR(x_j) = I(M; D)R_M(x_j) + \sum_{i=1}^{K} I(P_i; D)R_{P_i}(x_j), \tag{3}$$

where K is the number of applied proxies, and $I(\cdot; D)$ is the measured influence in the current iteration based on $G(\cdot; D)$. We use a softmax function with temperature to scale the sum of influence to 1 and use the temperature τ to control the softness of output distribution. The formulation is as follows,

$$I(\cdot; D) = \frac{exp(G(\cdot; D)/\tau)}{\sum exp(G(\cdot; D)/\tau)}, \ \tau = \frac{\tau_0}{1 + \log T}, \tag{4}$$

where τ_0 is the only hyper-parameter, and T is the current number of BO iteration. In each iteration, ProxyBO selects the configuration with the lowest combined ranking (CR) to evaluate. As there might be other ways to combine BO and proxies, we show that ProxyBO performs better than other intuitive combinations in the following section.

Algorithm Summary

Algorithm 1 illustrates the sampling procedure of ProxyBO. It first computes the generalization ability measurements G (Line 2) and converts them to influence I (Line 3). Then, it ranks the sampled configurations (Lines 4-6) and combines the rankings (Line 7). ProxyBO follows a typical BO framework by replacing the original EI-maximizing procedure with the combined ranking-minimizing procedure.

Discussions

In this subsection, we discuss the properties of our proposed ProxyBO as follows:

Extension. ProxyBO is independent of the choice of BO surrogate and zero-cost proxies, and users can replace those components with state-of-the-art ones in future researches. Moreover, it also supports other orthogonal methods for acceleration, like transfer learning.

Time Complexity. The time complexity of each iteration in ProxyBO is $\mathcal{O}(|D|log|D|)$, which is dominated by the cost of fitting a probabilistic random forest surrogate. Note that since the computation cost of proxy scores is constant during each iteration, it is not taken into account.

Overhead Analysis. During each iteration, the computation cost of proxy scores is QT_p in which Q is the number of computed configurations, and T_p is the average cost of computation. In practice, T_p can be calculated in seconds, and Q configurations can be parallelly computed even on a single GPU. We set Q to be 500, and the time cost for each iteration is less than a minute. Compared with the actual training cost (hours), this overhead can almost be ignored.

Convergence Discussion. As the number of observations grows, the proxies accumulate misrankings while the surrogate generalizes better. As a result, the generalization ability of the BO surrogate will gradually outperform the proxies. Meanwhile, the temperature τ declines when T grows, which sharpens the distribution and leads to the domination of the BO surrogate on influence. Finally, ProxyBO puts almost all the weights on the BO surrogate, and the algorithm reverts to standard BO, which enjoys a convergence guarantee (Hutter, Hoos, and Leyton-Brown 2011). We provide the analysis of when the combination function reverts to standard BO.

Assumption 1. *As proxies are not learning models, $G(P_i; D)$ is relatively stable and has an upper bound. Denote the least upper bound among all proxies as G_u; As a learning model, we expect that $G(M; D)$ will consistently outperform all proxies after sufficient rounds T_e, and then $G(M; D)$ has a lower bound G_l, where $G_l > G_u$.*

Theorem 1. *After $T_c = \max\{T_e, \exp(\tau_0 * \frac{\log KQ}{G_l - G_u} - 1))\}$ rounds where K, Q is the number of proxies and sampled configurations, ProxyBO reverts to standard BO.*

The proofs are provided in the appendix. We will also illustrate this trend of influence in the following section.

Difference with Previous Methods. While ProxyBO shares the same spirit as previous work (Li et al. 2021a; Feurer, Letham, and Bakshy 2018) that auxiliary information is applied to further improve BO using the normalized Kendall-Tau correlation, we discuss the difference between ProxyBO and previous methods as follows: **D1**: The first difference is that the auxiliary information is different (i.e., history surrogates in RGPE, low-fidelity surrogates in MFES, and zero-cost proxies in ProxyBO). **D2**: The main difference is how we combine the two parts. As proxies do not provide mean and variance outputs, we propose to combine them in acquisition function instead of surrogate as in RGPE and MFES. However, taking NAS-Bench-101 as an example, the challenges are that: 1) The values are of significantly different scales. The synflow values range from 10^{13} to 10^{17} while the jacob_cov values are around 272. 2) The distributions are also different. Most synflow values are close to 10^{14} while jacob_cov values are uniformly distributed. We notice that we only aim to choose the architecture with the largest acquisition value. Therefore, we convert the values into ranks so that they can be directly summed up. **D3**: We apply softmax with temperature to ensure that ProxyBO reverts to standard BO given sufficient iterations. The theoretical analysis is also provided above.

	snip	synflow	jacob_cov
NAS-Bench-101	-0.16 (-0.00)	0.37 (0.14)	-0.38 (-0.08)
NB2 CIFAR-10	0.60 (-0.36)	0.72 (0.12)	0.74 (0.15)
NB2 CIFAR-100	0.64 (-0.09)	0.71 (0.42)	0.76 (0.06)
NB2 ImageNet16-120	0.58 (0.13)	0.70 (0.55)	0.75 (0.06)
NAS-Bench-ASR	0.03 (0.13)	0.41 (-0.01)	-0.36 (0.06)

Table 1: Spearman ρ of proxies for all (and top-10%) architectures in NAS spaces.

Experiments and Results

To evaluate ProxyBO, we apply it on several public NAS benchmarks. Compared with state-of-the-art baselines, we list three main insights that we will investigate: 1) ProxyBO can dynamically measure the influence of zero-cost proxies during the search process; 2) ProxyBO can effectively integrate those proxies with the BO procedure without prior knowledge. In other words, good proxies greatly boost performance while bad ones influence little; 3) ProxyBO achieves promising results and can significantly accelerate neural architecture search. It reaches similar performance to other methods while spending much less search time.

Experimental Setup

Baselines. In the main experiment, we compare the proposed method ProxyBO with the following **fifteen** baselines — *Five regular methods:* (1) Random search (RS), (2) REINFORCE (RL) (Williams 1992), (3) Regularized evolutionary algorithm (REA) (Real et al. 2019), (4) Bayesian optimization (BO) (Hutter, Hoos, and Leyton-Brown 2011), (5) Binary relation predictor (BRP) (Dudziak et al. 2020) — *Two multi-fidelity methods:* (6) BOHB (Falkner, Klein, and Hutter 2018), (7) MFES (Li et al. 2021a), — *Two weight-sharing methods:* (8) DARTS-PT (Wang et al. 2021), (9) ENAS (Pham et al. 2018), — *Three zero-cost proxies:* (10) Snip (Lee, Ajanthan, and Torr 2018), (11) Synflow (Tanaka et al. 2020), (12) Jacob_cov (Mellor et al. 2021), — *Three zero-cost proxy-based methods:* (13) Warm-up BRP (Abdelfattah et al. 2021): BRP-NAS with warm-start based on the relative rankings of proxy scores. (14) A-REA (Mellor et al. 2021): REA that evaluates population with the largest proxy scores. (15) OMNI (White et al. 2021): BO using the proxy scores as inputs.

Benchmarks. To ensure reproducibility as in previous work (Abdelfattah et al. 2021), we conduct the experiments on four public NAS benchmarks: NAS-Bench-101 (Ying et al. 2019), NAS-Bench-201 (Dong and Yang 2020), NAS-Bench-ASR (Mehrotra et al. 2021), and NAS-Bench-301 (Siems et al. 2020) with the **real-world DARTS search space**. Detailed descriptions are provided in the appendix.

We demonstrate the Spearman ρ of proxy scores related to the test results of all models and top-10% models on each task in Table 1 on five tasks, and note that ρ is the prior knowledge that can not be obtained before optimization. We observe that no proxy dominates the others on all the tasks, and some correlation coefficients are even negative, leading to a number of incorrect rankings.

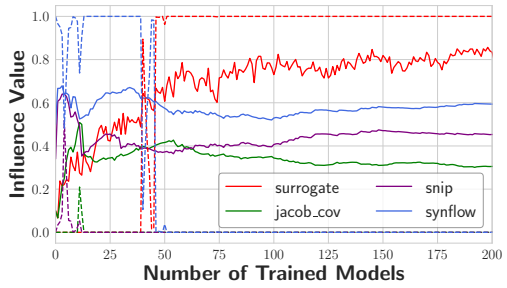


Figure 2: The solid and dash lines refer to generalization ability measurements and influence values.

(a) NAS-Bench-101 (Optimal: 5.68%)

Method	None	snip	jacob_cov	synflow	all
A-REA	6.19	6.27	6.62	5.95	6.07
OMNI (PRF)	6.14	6.28	6.37	5.91	6.05
Warm-up BO (PRF)	6.14	6.31	6.30	5.91	6.08
ProxyBO (PRF)	6.14	**6.18**	**6.14**	**5.87**	**5.96**

(b) NB2 ImageNet-16-120 (Optimal: 52.69%)

Method	None	snip	jacob_cov	synflow	all
A-REA	53.08	53.43	53.15	52.80	52.91
OMNI (PRF)	53.12	53.21	53.07	52.83	52.92
Warm-up BO (PRF)	53.12	53.24	53.20	52.77	53.08
ProxyBO (PRF)	53.12	**53.18**	**52.93**	**52.74**	**52.82**

Table 2: Mean test errors (%) of zero-cost proxy-based methods with different proxies.

Basic Settings. In the following subsections, we report the average best test error for NAS-Bench-101, 201, 301 and the average best test Phoneme error rate (PER) for NAS-Bench-ASR. The "average best" refers to the best-observed performance during optimization, which is non-increasing. For evaluation-based methods, we set the time budget as 200 times **the average cost of an entire evaluation** on each task. For zero-cost proxies, we randomly sample 1000 architectures and report the model with the best proxy score. Each weight-sharing method is run until convergence. To avoid randomness, weight-sharing methods are repeated 5 times, and the other methods are repeated 30 times. We plot the mean $\pm$ std. results in the following figures.

Implementation Details. We implement ProxyBO based on OpenBox (Li et al. 2021b), a toolkit for black-box optimization. The other baselines are implemented following their original papers. While GPWL and BRP contain specific implementations for each benchmark, we only evaluate them on their supported benchmarks based on the open-source version. Furthermore, we use Pytorch (Paszke et al. 2019) to train neural networks and calculate proxy scores. The population size for REA and A-REA is 20; in BRP and Warm-up BRP, 30 models are sampled to train the predictor during each iteration; in Warm-up BRP, we randomly sample 256 models to perform a warm start; if not mentioned, the zero-cost proxy-based methods apply **all** three proxies. The η and R are set to 3 and 27 in BOHB and MFES, respectively. The τ_0 in ProxyBO is set to 0.05. In the last experiment using NAS-Bench-301, we use "xgb" as the performance predictor and "lgb_runtime" as the runtime predictor. The experiments are conducted on a machine with 64 'AMD EPYC 7702P' CPU cores and two 'RTX 2080Ti' GPUs.

Empirical Analysis

ProxyBO can measure the influence of proxies during the search process. To show the influence of different proxies, we demonstrate the trend of generalization ability measurements G (solid lines) and influence values I (dash lines) on NAS-Bench-101 in Figure 2. As shown in Table 1, the correlation coefficients of `snip` and `jacob_cov` are negative. Therefore, their measurements are much lower than `synflow`'s, and their influence values are kept as zero after about 13 evaluations. `synflow`, the best among the three proxies, shows the largest influence in the beginning 36 evaluations. Our result is also consistent with previous work (Ning et al. 2021) that using the best proxy performs better than a voting ensemble. However, since zero-cost proxies are formulated metrics, their measurements are relatively stable. The measurement of BO surrogate exceeds `synflow` at the 39-th evaluation, and it takes the surrogate another 10 evaluations to enlarge the gap. After that, the generalization ability of the surrogate further increases with more evaluations, and its influence keeps closely to 1. In this case, ProxyBO turns back to standard BO with a convergence guarantee.

ProxyBO can effectively integrate BO procedure with zero-cost proxies. To show that combining BO and proxies are non-trivial, we compare different methods of using zero-proxy proxies. Based on probabilistic random forest (PRF), we add the baselines OMNI (White et al. 2021) and Warm-up BO with 40 start points of good proxy scores. While BO (PRF) can not be directly extended to support moving proposal, we use A-REA as the baseline. The results of using different proxies are presented in Table 2. We find that: 1) ProxyBO outperforms Warm-up and OMNI. The reason is that, Warm-up applies a static setting rather than the dynamic one in ProxyBO, and the different distribution of proxy scores across tasks makes it difficult to rescale the scores as inputs for OMNI. 2) If the correlations are not accessible, ProxyBO performs the best among baselines using all proxies. When using bad proxies with negative correlation (see Table 1), ProxyBO performs similar to that without proxies. If we know the best correlated proxy (synflow) beforehand, ProxyBO saves budget for identifying bad proxies. It reduces the regret (i.e., the distance to the global optima) of the second-best method by **17-38%** and performs better than using all proxies. We provide additional ablation study, effectiveness analysis, and discussions on how to select τ_0 based on prior knowledge in the appendix.

ProxyBO achieves promising results. Table 3 shows the test results on five tasks given the budget of 200 evaluations. We observe that zero-cost proxies require extremely short computation time, but the final results are not satisfactory. The reason is that it can not correctly rank the most-accurate architectures (see top-10% architectures in Table 1). Note that, though weight-sharing methods and zero-cost proxies require

Method	Runtime (#Eval)	NAS-Bench-101	NB2-CIFAR-10	NB2-CIFAR-100	NB2-ImageNet16-120	NAS-Bench-ASR
Regular Methods						
RS	200	6.34 ± 0.12	9.11 ± 0.21	27.97 ± 0.66	54.01 ± 0.55	21.61 ± 0.10
RL	200	6.31 ± 0.14	9.02 ± 0.24	27.66 ± 0.65	53.58 ± 0.45	21.62 ± 0.09
REA	200	6.19 ± 0.24	8.62 ± 0.21	26.67 ± 0.35	53.08 ± 0.36	21.50 ± 0.07
BO	200	6.14 ± 0.23	8.80 ± 0.22	27.03 ± 0.45	53.12 ± 0.37	21.47 ± 0.06
BRP	200	6.05 ± 0.16	8.58 ± 0.13	26.58 ± 0.12	52.96 ± 0.29	21.50 ± 0.08
Multi-fidelity Methods						
BOHB	200	6.26 ± 0.18	8.95 ± 0.28	27.65 ± 0.72	53.77 ± 0.53	21.74 ± 0.16
MFES	200	6.10 ± 0.17	8.71 ± 0.18	26.57 ± 0.13	53.19 ± 0.18	21.71 ± 0.12
Weight-sharing Methods						
ENAS	$\approx$7	8.17 ± 0.42	46.11 ± 0.58	86.04 ± 2.33	85.19 ± 2.10	24.45 ± 0.90
DARTS-PT	$\approx$18	7.79 ± 0.61	15.33 ± 2.23	34.03 ± 2.24	61.36 ± 1.91	24.08 ± 0.43
Zero-cost Proxies						
Snip	$<$1	10.68 ± 2.16	13.45 ± 1.80	36.41 ± 3.36	71.94 ± 9.09	31.61 ± 18.17
Jacob_cov	$<$1	13.86 ± 1.86	12.19 ± 1.60	32.99 ± 2.84	60.43 ± 4.46	69.95 ± 24.67
Synflow	$<$1	8.32 ± 1.64	10.30 ± 0.94	29.55 ± 1.77	56.94 ± 3.57	25.70 ± 12.91
Zero-cost Proxy-based Methods						
A-REA	200	6.07 ± 0.21	$\mathbf{8.54 \pm 0.08}$	26.59 ± 0.11	52.91 ± 0.24	21.47 ± 0.06
Warm-up BRP	200	6.04 ± 0.15	8.58 ± 0.21	26.60 ± 0.31	53.02 ± 0.35	21.51 ± 0.10
OMNI	200	6.05 ± 0.11	8.60 ± 0.14	26.64 ± 0.29	52.92 ± 0.26	21.48 ± 0.05
ProxyBO	200	$\mathbf{5.96 \pm 0.13}$	$\mathbf{8.54 \pm 0.10}$	$\mathbf{26.52 \pm 0.17}$	$\mathbf{52.82 \pm 0.19}$	$\mathbf{21.43 \pm 0.03}$
Optimal	/	5.68	8.48	26.49	52.69	21.40

Table 3: Mean $\pm$ std. test errors (%) on NAS-Bench-101 and NAS-Bench-201, and test PERs (%) on NAS-Bench-ASR. "NB2" refers to NAS-Bench-201, and "Optimal" refers to the ground-truth optima. The evaluation number of multi-fidelity methods, weight-sharing methods, and zero-cost proxies is computed by their runtime divided by the average training time of architectures.

less budget than evaluation-based methods, their performance has converged. In addition, since the multi-fidelity methods conduct a large number of evaluations with fewer epochs, they train fewer models to convergence than the standard BO, which may lead to an under-estimation of converged model performance given limited budget. In our experiments, we observe that MFES performs worse than BO on NB2-ImageNet16-120 and NAS-Bench-ASR. Finally, due to the limitations of the warm-up strategy, Warm-up BRP slightly outperforms BRP on NAS-Bench-101 but performs worse on the other benchmarks. While A-REA can not distinguish bad proxies, it performs well on NB2 where each proxy is effective but less competitive on other benchmarks. Among the competitive baselines, ProxyBO achieves the best average test results on all five tasks and remarkablely, it reduces the test regret of the best baseline by **70%** on NB2-CIFAR-100.

We also apply ProxyBO to state-of-the-art surrogate GPWL (Ru et al. 2021) on NAS-Bench-201 and NAS-Bench-101. The corresponding results are shown in Table 5. All methods use the same strategy to sample candidates for fair comparison. We find that ProxyBO (GPWL) achieves the best results on four tasks. Since using GPWL alone is empirically superior to PRF, the gain of applying ProxyBO is smaller on GPWL than on PRF. But it still decreases the regret of BO (GPWL) by **25-84%** on four tasks, which indicates the effectiveness of ProxyBO on utilizing useful zero-cost proxies.

In addition, we compare ProxyBO with SOTA weight-sharing method β-DARTS (Ye et al. 2022) on NB2-ImageNet16-120. ProxyBO may not obtain a satisfactory result as fast as weight-sharing methods due to initialization. It takes β-DARTS and ProxyBO 15 and 28 GPU hours to obtain the same error (53.66%). However, when given a larger budget, the error of ProxyBO continuously decreases while β-DARTS has already converged. The error decrease of ProxyBO over β-DARTS given different times of budget are shown in Table 4. While the results of ProxyBO (16x, 52.94%) approach the optimum (52.69%), this improvement is relatively considerable. Users can decide whether to use ProxyBO and how much budget to offer by balancing the tradeoff between time and performance. More detailed discussion and results are provided in the appendix.

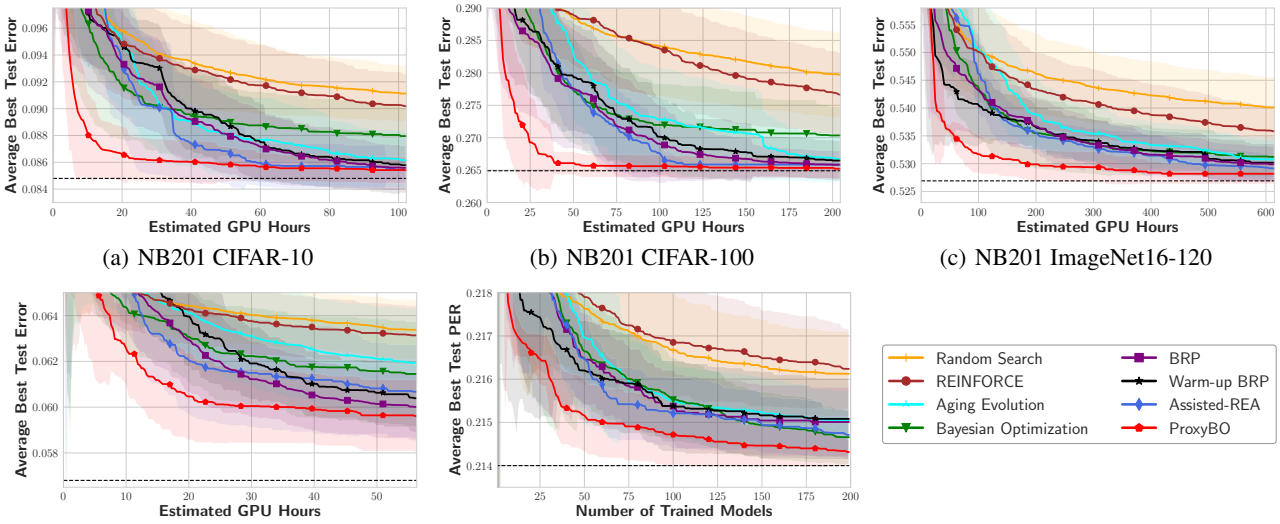
	1.87x	4x	8x	16x	32x
Gap	0.00	0.22	0.53	0.72	0.85

Table 4: Error decrease (%) when given more time budget compared with β-DARTS on NB2-ImageNet16-120.

Method	NB101	NB2 C10	NB2 C100	NB2 I16
BO (PRF)	6.14	8.80	27.03	53.12
ProxyBO (PRF)	5.96	8.54	26.52	52.82
BO (GPWL)	5.88	8.56	26.55	52.97
ProxyBO (GPWL)	**5.83**	**8.52**	**26.50**	**52.78**
Optimal	5.68	8.48	26.49	52.69

Table 5: Mean test errors (%) with different surrogates.

ProxyBO can significantly accelerate NAS. Figure 3 demonstrates the search results of regular and zero-cost



(a) NB201 CIFAR-10 (b) NB201 CIFAR-100 (c) NB201 ImageNet16-120

(d) NAS-Bench-101 (e) NAS-Bench-ASR

Figure 3: Test results during neural architecture search on three benchmarks. The black dash lines refer to the global optima.

	BRP	W-BRP	A-REA	ProxyBO
NAS-Bench-101	94	105	74	**41**
NB2 CIFAR-10	170	178	108	**44**
NB2 CIFAR-100	142	184	98	**37**
NB2 ImageNet16-120	144	168	144	**46**
NAS-Bench-ASR	179	213	177	**51**

Table 6: Number of evaluations required to achieve the same average results as REA with 200 evaluations.

proxy-based methods. Consistent with Table 3, we observe that Warm-up BRP only shows acceleration in early rounds on NB2 ImageNet16-120 and NAS-Bench-ASR. Compared with weight-sharing methods and zero-cost proxies, it takes ProxyBO less than 8 evaluations to surpass their converged results. In addition, the test results of ProxyBO decrease rapidly before 75 iterations, and it consistently outperforms the other baselines on the five tasks. To show the speedup of ProxyBO, we further compare the number of trained models required to achieve the same average results as REA using 200 evaluations in Table 6. Concretely, ProxyBO achieves $\mathbf{3.92 - 5.41}\times$ and $\mathbf{2.29 - 3.86}\times$ speedups relative to the state-of-the-art method REA and BRP, respectively.

We also evaluate ProxyBO on realistic DARTS search space using NAS-Bench-301. While BANANAS (White, Neiswanger, and Savani 2021) is a state-of-the-art BO method on that space, we apply ProxyBO to BANANAS and compare it with evaluation-based methods, and the results are shown in Table 7. As previous study (White et al. 2021) claims that local search is not so effective on large space given limited budget, REA requires about twice the budget as BANANAS. Among compared methods, ProxyBO outperforms A-REA, and it spends $\mathbf{0.70}\times$ GPU hours to achieve the same results as BANANAS. More detailed results on NAS-Bench-301 are provided in the appendix.

	RS	REA	A-REA	BANANAS	ProxyBO
#Eval	≈2k	441	304	200	**142**
GPU Hours	≈3k	663	424	289	**201**

Table 7: Number of evaluations required to achieve the same results as BANANAS (5.10%).

In addition, we evaluate BANANAS and ProxyBO (BANANAS) on DARTS space using CIFAR10. The search takes nearly 3 days, and the final architecture is trained given 600 epochs. To save evaluation costs during searching, we train 8-cell architectures instead of 20-cell ones for 40 epochs. The final test errors of BANANAS and ProxyBO are 2.75% and 2.68%, respectively, which show that ProxyBO can further improve BANANAS.

Conclusion

In this paper, we introduced ProxyBO, an efficient Bayesian optimization framework that leverages the auxiliary knowledge from zero-cost proxies. In ProxyBO, we proposed two components, namely the generalization ability measurement and dynamic influence combination, which tackles the unforeseeable reliability and one-shot usage issues in existing methods, and developed a more principled way to utilize zero-cost proxies. We evaluated ProxyBO on four public benchmarks and demonstrated its superiority over competitive baselines.

Acknowledgements

This work is supported by NSFC (No. 61832001 and U22B2037) and Kuaishou-PKU joint program. Wentao Zhang and Bin Cui are the corresponding authors. Special thanks to Tianyi Bai and Yupeng Lu for their help during the rebuttal period.

References

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

Abraham, S. S.; et al. 2021. FairLOF: Fairness in Outlier Detection. *Data Science and Engineering*, 6(4): 485–499.

Cai, H.; Chen, T.; Zhang, W.; Yu, Y.; and Wang, J. 2018. Efficient architecture search by network transformation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Cao, Y.; and Fleet, D. J. 2014. Generalized product of experts for automatic and principled fusion of Gaussian process predictions. *arXiv preprint arXiv:1410.7827*.

Chen, W.; Gong, X.; and Wang, Z. 2021. Neural Architecture Search on ImageNet in Four GPU Hours: A Theoretically Inspired Perspective. In *International Conference on Learning Representations*.

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

Dudziak, L.; Chau, T.; Abdelfattah, M.; Lee, R.; Kim, H.; and Lane, N. 2020. BRP-NAS: Prediction-based NAS using GCNs. *Advances in Neural Information Processing Systems*, 33.

Falkner, S.; Klein, A.; and Hutter, F. 2018. BOHB: Robust and efficient hyperparameter optimization at scale. In *International Conference on Machine Learning*, 1437–1446. PMLR.

Feurer, M.; Letham, B.; and Bakshy, E. 2018. Scalable meta-learning for bayesian optimization using ranking-weighted gaussian process ensembles. In *AutoML Workshop at ICML*.

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

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

Hutter, F.; Hoos, H. H.; and Leyton-Brown, K. 2011. Sequential model-based optimization for general algorithm configuration. In *International Conference on Learning and Intelligent Optimization*, 507–523. Springer.

Jones, D. R.; Schonlau, M.; and Welch, W. J. 1998. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4): 455–492.

Kandasamy, K.; Neiswanger, W.; Schneider, J.; Póczos, B.; and Xing, E. P. 2018. Neural architecture search with Bayesian optimisation and optimal transport. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2020–2029.

Krishnakumar, A.; White, C.; Zela, A.; Tu, R.; Safari, M.; and Hutter, F. 2022. NAS-Bench-Suite-Zero: Accelerating Research on Zero Cost Proxies. *arXiv preprint arXiv:2210.03230*.

Lee, H.; Hyung, E.; and Hwang, S. J. 2021. Rapid neural architecture search by learning to generate graphs from datasets. *arXiv preprint arXiv:2107.00860*.

Lee, N.; Ajanthan, T.; and Torr, P. 2018. Snip: Single-shot Network Pruning Based on Connection Sensitivity. In *International Conference on Learning Representations*.

Li, L.; Jamieson, K.; DeSalvo, G.; Rostamizadeh, A.; and Talwalkar, A. 2017. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1): 6765–6816.

Li, Y.; Shen, Y.; Jiang, H.; Zhang, W.; Li, J.; Liu, J.; Zhang, C.; and Cui, B. 2022a. Hyper-Tune: Towards Efficient Hyperparameter Tuning at Scale. *arXiv preprint arXiv:2201.06834*.

Li, Y.; Shen, Y.; Jiang, J.; Gao, J.; Zhang, C.; and Cui, B. 2021a. MFES-HB: Efficient Hyperband with Multi-Fidelity Quality Measurements. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 8491–8500.

Li, Y.; Shen, Y.; Zhang, W.; Chen, Y.; Jiang, H.; Liu, M.; Jiang, J.; Gao, J.; Wu, W.; Yang, Z.; Zhang, C.; and Cui, B. 2021b. OpenBox: A Generalized Black-box Optimization Service. *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*.

Li, Y.; Shen, Y.; Zhang, W.; Zhang, C.; and Cui, B. 2022b. VolcanoML: speeding up end-to-end AutoML via scalable search space decomposition. *The VLDB Journal*, 1–25.

Liu, H.; Simonyan, K.; and Yang, Y. 2019. DARTS: Differentiable Architecture Search. In *International Conference on Learning Representations*.

Ma, L.; Cui, J.; and Yang, B. 2019. Deep neural architecture search with deep graph bayesian optimization. In *2019 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, 500–507. IEEE.

Mehrotra, A.; Ramos, A. G. C.; Bhattacharya, S.; Dudziak, Ł.; Vipperla, R.; Chau, T.; Abdelfattah, M. S.; Ishtiaq, S.; and Lane, N. D. 2021. NAS-Bench-ASR: Reproducible Neural Architecture Search for Speech Recognition. In *International Conference on Learning Representations*.

Mellor, J.; Turner, J.; Storkey, A.; and Crowley, E. J. 2021. Neural architecture search without training. In *International Conference on Machine Learning*, 7588–7598. PMLR.

Ning, X.; Tang, C.; Li, W.; Zhou, Z.; Liang, S.; Yang, H.; and Wang, Y. 2021. Evaluating Efficient Performance Estimators of Neural Architectures. *Advances in Neural Information Processing Systems*, 34.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32: 8026–8037.

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.

Real, E.; Aggarwal, A.; Huang, Y.; and Le, Q. V. 2019. Regularized evolution for image classifier architecture search. In *Proceedings of the 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*.

Shu, Y.; Cai, S.; Dai, Z.; Ooi, B. C.; and Low, B. K. H. 2021. NASI: Label-and Data-agnostic Neural Architecture Search at Initialization. *arXiv preprint arXiv:2109.00817*.

Shu, Y.; Dai, Z.; Wu, Z.; and Low, B. K. H. 2022. Unifying and Boosting Gradient-Based Training-Free Neural Architecture Search. *arXiv preprint arXiv:2201.09785*.

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*.

Snoek, J.; Larochelle, H.; and Adams, R. P. 2012. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, 2951–2959.

So, D.; Le, Q.; and Liang, C. 2019. The evolved transformer. In *International Conference on Machine Learning*, 5877–5886. PMLR.

Springenberg, J. T.; Klein, A.; Falkner, S.; and Hutter, F. 2016. Bayesian optimization with robust Bayesian neural networks. *Advances in neural information processing systems*, 29: 4134–4142.

Tanaka, H.; Kunin, D.; Yamins, D. L.; and Ganguli, S. 2020. Pruning neural networks without any data by iteratively conserving synaptic flow. *Advances in Neural Information Processing Systems*, 33.

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. In *International Conference on Learning Representations*.

Wang, R.; Cheng, M.; Chen, X.; Tang, X.; and Hsieh, C.-J. 2021. Rethinking Architecture Selection in Differentiable NAS. In *International Conference on Learning Representations*.

Wei, T.; Wang, H.; Tu, W.; and Li, Y. 2022. Robust model selection for positive and unlabeled learning with constraints. *Science China Information Sciences*, 65(11): 1–13.

White, C.; Neiswanger, W.; and Savani, Y. 2021. BANANAS: Bayesian Optimization with Neural Architectures for Neural Architecture Search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 10293–10301.

White, C.; Zela, A.; Ru, B.; Liu, Y.; and Hutter, F. 2021. How Powerful are Performance Predictors in Neural Architecture Search? *arXiv preprint arXiv:2104.01177*.

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

Xu, J.; Zhao, L.; Lin, J.; Gao, R.; Sun, X.; and Yang, H. 2021. KNAS: green neural architecture search. In *International Conference on Machine Learning*, 11613–11625. PMLR.

Yang, L.; Zhang, Z.; and Hong, S. 2022. Diffusion Models: A Comprehensive Survey of Methods and Applications. *arXiv preprint arXiv:2209.00796*.

Ye, P.; Li, B.; Li, Y.; Chen, T.; Fan, J.; and Ouyang, W. 2022. b-DARTS: Beta-Decay Regularization for Differentiable Architecture Search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10874–10883.

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

Zhang, W.; Jiang, J.; Shao, Y.; and Cui, B. 2020. Snapshot boosting: a fast ensemble framework for deep neural networks. *Science China Information Sciences*, 63(1): 1–12.

Zhang, W.; Shen, Y.; Lin, Z.; Li, Y.; Li, X.; Ouyang, W.; Tao, Y.; Yang, Z.; and Cui, B. 2022. Pasca: A graph neural architecture search system under the scalable paradigm. In *Proceedings of the ACM Web Conference 2022*, 1817–1828.

Zhou, D.; Zhou, X.; Zhang, W.; Loy, C. C.; Yi, S.; Zhang, X.; and Ouyang, W. 2020. Econas: Finding proxies for economical neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11396–11404.

Zhu, D.-H.; Dai, X.-Y.; and Chen, J.-J. 2021. Pre-Train and Learn: Preserving Global Information for Graph Neural Networks. *Journal of Computer Science and Technology*, 36(6): 1420–1430.

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