UP-NAS: Unified Proxy for Neural Architecture Search

Yi-Cheng Huang[1*], Wei-Hua Li[1*], Chih-Han Tsou[1], Jun-Cheng Chen[2], Chu-Song Chen[1]
[1]National Taiwan University, [2]Academia Sinica
{r10922136, d12922009, r10944039}@csie.ntu.edu.tw,
pullpull@citi.sinica.edu.tw, chusong@csie.ntu.edu.tw

Abstract

Recently, zero-cost proxies for neural architecture search (NAS) have attracted increasing attention. They allow us to discover top-performing neural networks through architecture scoring without requiring training a very large network (i.e., supernet). Thus, it can save significant computation resources to complete the search. However, to our knowledge, no single proxy works best for different tasks and scenarios. To consolidate the strength of different proxies and to reduce search bias, we propose a unified proxy neural architecture search framework (UP-NAS) which learns a multi-proxy estimator for predicting a unified score by combining multiple zero-cost proxies. The predicted score is then used for an efficient gradient-ascent architecture search in the embedding space of the neural network architectures. Our approach can not only save computational time required for multiple proxies during architecture search but also gain the flexibility to consolidate the existing proxies on different tasks. We conduct experiments on the search spaces of NAS-Bench-201 and DARTS in different datasets. The results demonstrate the effectiveness of the proposed approach. Code is available at https://github.com/AI-Application-and-Integration-Lab/UP-NAS.

1. Introduction

Neural architecture search (NAS) is an automated machine learning method that aims to find optimal model structures by searching the neural network architecture space. Traditional deep learning models require experts to design the model structure. NAS simplifies this process by automatically exploring the search space to generate architectures that are better than those designed manually.

Existing NAS approaches can generally be divided into three categories: multi-shot NAS, one-shot NAS, and zero-shot NAS. Multi-shot NAS involves training multiple candidate architectures, which is time-consuming. To alleviate this issue, one-shot NAS was proposed and has become a more commonly adopted approach. It constructs a big model called supernet which is the union of the architecture search space. After training the supernet and leveraging some search strategies, we can obtain the optimal architecture. One-shot NAS can be roughly categorized into two groups: Differentiable NAS and Single-path NAS. Differentiable NAS, such as DARTS [18] and β-DARTS [41], use gradient descent algorithms to train a supernet, obtaining optimal architecture directly after training. Single-path NAS, such as SPOS [9] and ENAS [25], randomly chooses a path in the supernet and only updates the weights along the path. By employing these NAS solutions, we can obtain better-performed architectures after some searching mechanisms like Random Search or Evolutionary Search.

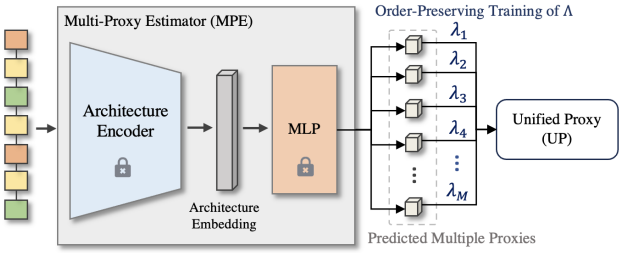


Figure 1. The illustration of the architecture of the proposed Multi-Proxy Estimator (MPE) and the Unified Proxy (UP). The proposed MPE consists of a pretrained Architecture Encoder and MLP. Moreover, the MPE and UP provide a novel and effective way to consolidate the strengths of multiple zero-cost proxies for improved neural architecture search.

However, these methods still require a large amount of computational resources and longer training time. Therefore, zero-cost proxy (or training-free NAS) has emerged as a promising technique to improve the efficiency and scalability of NAS. It has the potential to enable more complex and comprehensive searches of the neural architecture space. Previous works [1, 17, 20, 23] have demonstrated the effectiveness of zero-cost proxy in enhancing the search efficiency of various NAS methods. Although it lessens the computational burden, a main issue of the current training-

*Equal Contribution.

free NAS is that, despite various zero-cost proxies have been proposed, each proxy performs well for only specific cases [12]. A recent study in [2] learns to compose a set of operations based on an existing NAS benchmark to form the zero-cost proxy. Nevertheless, the method only provides rank-correlation performance on the adopted NAS benchmark without providing evidence of its generalization ability to other search spaces or datasets.

Since no single proxy is useful for all benchmark data, we propose to ensemble them to form the Unified Proxy (UP). To enlarge the resolution of the architecture search space and make the architectures compactly accessible, we propose a framework to explore the use of multiple proxies to further improve the accuracy and rank correlation of NAS. The main idea is to predict the representative proxy score of a given network architecture. To achieve this goal, we leverage the multiway learning paradigm to simultaneously predict the scores of multiple proxies and to consolidate them. For the shared architecture representation, we first train an autoencoder that can effectively represent an architecture as an embedding using its encoder, and then reconstruct the original architecture from the embedding using its decoder similar to arch2vec [40]. Then, the learned autoencoder allows us to express an architecture as a compact and differentiable embedding representation which can be concatenated with other network models for NAS. In our method, we concat architecture encoder with another multilayer perceptron (MLP) network as a surrogate model to form a Multi-Proxy Estimator (MPE) and predict the UP score, which is a composite of multiple proxy scores. At last, we employ the gradient ascent technique to find out the high-UP-score architecture. Through experiments using the search space of NAS-Bench-201 [8] (on CIFAR-10, CIFAR-100, and ImageNet-16-120) and DARTS [18] (on ImageNet), the results show that UP-NAS is able to search comparable architecture to other state-of-the-art NAS methods while having faster search speed than the evolutionary and random search algorithms. Specifically, when applying our framework to the searched architecture on DARTS-CIFAR-10 with the proposed weighted proxies and selected proxies, we can achieve SOTA error rates of 23.5% on the DARTS search space for ImageNet.

Main contributions of the proposed method are two-fold. First, the proposed Multi-Proxy Estimator (MPE) can faithfully predict the scores of each zero-cost proxy and combine the each proxy into an enhanced score. Next, MPE allows a more effective and faster search of the optimal architecture through gradient ascent operations than traditional random-based and evolutionary-based search methods.

2. Related Work

Due to a large number of related works in the literature, we briefly introduce the recent relevant approaches below.

2.1. Predictor-based NAS

There are several existing NAS methods that utilize an external predictor to predict the performance of a neural architecture before training. NAO [21] converts discrete architecture encoding space into a continuous embedding space using a pair of architecture encoder and decoder, and conducts architecture search on the embedding space with the guidance of a performance predictor. To be specific, they utilize the performance predictor reversely such that architecture embedding is modified according to its gradient. BRP-NAS [5] uses Graph Convolution Network (GCN) to extract features from an architecture encoding with a binary relation aware training approach. NPENAS [35] enhances the exploration ability of conventional evolutionary approach by incorporating a neural predictor to guide the architecture search. SemiAccessor [31] and SemiNAS [22] are methods that use semi-supervised learning to train the predictor. GMAE [10] proposes a self-supervised method to pretrain the predictor with untrained architectures for reducing the dependence on supervision data and performance enhancement. FBNetV3 [7] utilizes a surrogate model to estimate the performance of an architecture with the consideration of the training hyperparameters, allowing a simultaneous search of optimal architecture and its training hyperparameters. NAR-Former [42] applies a tokenizer to encode operation and topology information of the neural network into sequence and adapts a multi-stage fusion transformer to learn the vector representation from the sequence.

As compared to NAO which heavily relies on the ground-truth validation accuracies for training, our approach leverages arch2vec [40], learning an architecture autoencoder to extract an architecture embedding in an unsupervised manner. In addition, we propose a novel multi-proxy estimator to consolidate multiple zero-cost proxy scores. This allows the proposed approach to conduct a more unbiased architecture performance estimation and to enable a precise and compact architecture search through the gradient ascent strategy.

2.2. Zero-Cost Proxies

Zero-Cost Proxies aim to accurately calculate the quality of an architecture without the need for training. The existing proxies can be divided into two classes: gradient-based and gradient-free based [15].

Gradient-based Proxy. This type of proxies attempts to measure the parameter importance through gradients. Grad-norm [1] calculates the sum for each layer's gradient, defined as $Grad(\theta) = |\frac{\partial L}{\partial \theta}|$, where L is the loss function of a network with parameters θ. Snip [14] multiplies the value of the parameter and the corresponding gradient to measure importance both in forward and backward propagation, which is defined as $Snip(\theta) = |\frac{\partial L}{\partial \theta} \odot \theta|$. Synflow [30] computes a loss simply to be the product of all

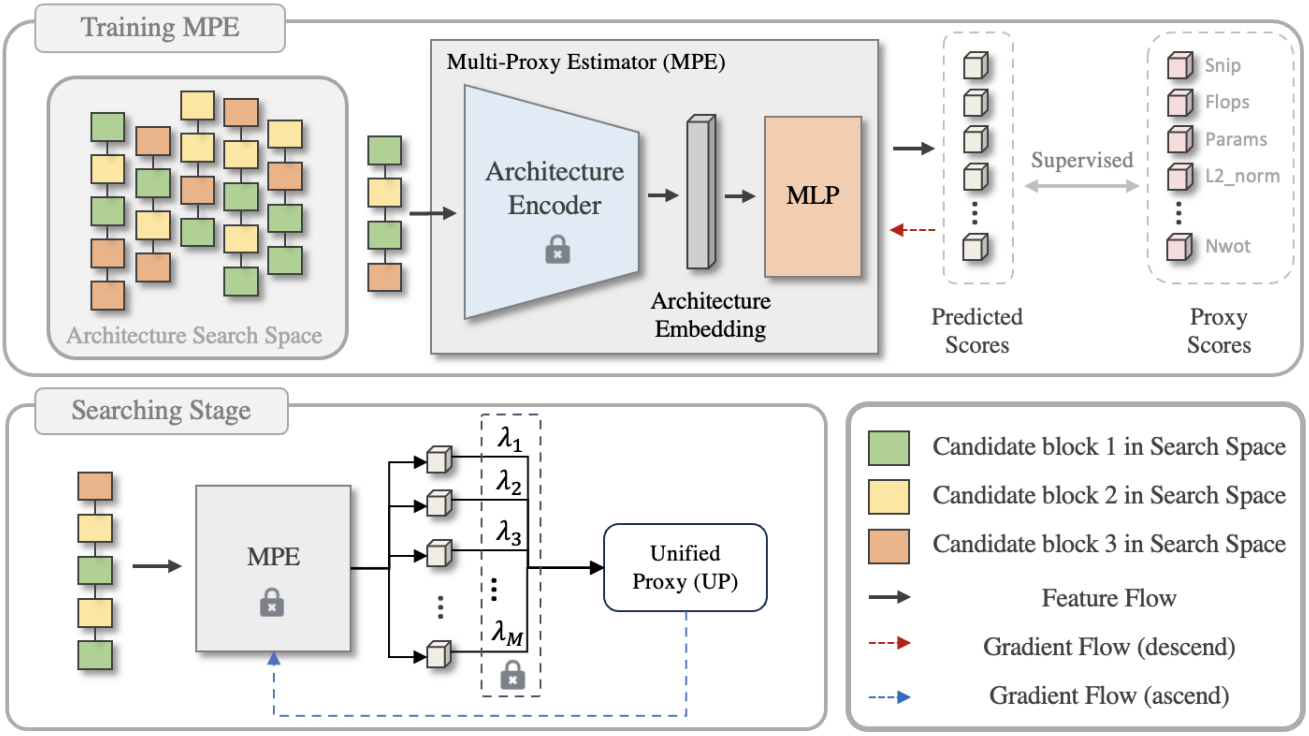


Figure 2. The proposed Unified-Proxy Neural Architecture Search (UP-NAS) where we introduce a Multi-Proxy Estimator (MPE) and Unified Proxy (UP) score. The details can be found in the Section 3.

parameters in the network. Similar to SNIP, it is defined as $Synflow(\theta) = \frac{\partial L}{\partial \theta} \odot \theta$. Grasp [33] approximates the change in gradient norm. It calculates the product of the Hessian matrix and the gradients, defined as $Grasp(\theta) = -(H\frac{\partial L}{\partial \theta}) \odot \theta$, where H is the hessian matrix of the network. Fisher information [19, 32] can measure the importance of the parameters in a network. Jacobian Covariant [23] is another proxy that reflects the expressiveness of deep networks. Zen-Score [17] is a metric that quantifies the expressivity of a deep neural network by averaging the Gaussian complexity of linear function in each linear region.

Gradient-free Proxy. In NASWOT [23], a simple algorithm is proposed for searching powerful neural networks without training by incorporating a measure of activation overlaps between datapoints in untrained networks. In EPE-NAS [20], the authors introduce an efficient performance estimation strategy that scores untrained networks by analyzing intra and inter-class correlations.

In our work, we adopt the implementation of zero cost proxies by NASLib [12, 26], including *grad-norm* [1], *snip* [14], *synflow* [30], *grasp* [33], *fisher* [32], *jacov* [23], *zen-score* [17], *nwot* [23], *epe-nas* [20], *flops* [24], *l2-norm* [1], *params* [24] and *plain* [1].

Currently, several NAS methods have already utilized zero-cost proxies to assist their search process. TE-NAS [6] is a training-free NAS that iteratively prunes paths from a supernet, according to their ranking relationship of two zero-cost metrics. ProxyBO [28] develop a sophisticated approach to combine and dynamically adjust the weight of existing zero-cost proxies, and use Bayesian Optimization for search. ZeroCost-PT [36] improves DARTS-PT [34] by substituting zero-cost proxy for validation accuracy. Recently, Sun *et al.* [29] propose ξ-based gradient signal-to-noise ratio (ξ-GSNR), combining with pruning/evolutionary algorithm to search better architectures. DELE [43] is a predictor-based method that "warms up" the predictor before training it with the actual performance to circumvent the need for a large amount of expensive label data.

Our method simply combines different zero-cost proxies by adding them with a set of fixed combination weights, which is easily scalable and effective.

3. Method

In this section, we introduce UP-NAS, a novel gradient-ascent learning framework that utilizes a unified zero-cost proxy score to search for an architecture. Our unified proxy is a combination of $\mathcal{M}$ zero-cost proxies. The combination coefficients $\Lambda = [\lambda_1, \lambda_2 ... \lambda_M]$ are learned based on a small existing NAS-benchmark (eg., NAS-Bench-201-CIFAR-10) with an order-preserving loss, and then applied to all other unseen architecture spaces. To avoid exhausted

evaluations for the $\mathcal{M}$ zero-cost proxies of the architectures encountered in the search process every time, we propose MPE, an estimator for the multiway zero-cost proxies. It consists of an architecture-to-vector (arc2vec) encoder network and a multi-layer perceptron (MLP), which estimates the scores obtained by the $\mathcal{M}$ proxies, respectively, of the input architecture. More details of the MPE module are illustrated in Figure 2.

3.1. Problem Formulation

A proxy is defined as a function that maps an architecture to a real number: $f_{zc} : \mathcal{A} \rightarrow \mathbf{R}$. Studies have found a significant relationship between several zero cost proxies and the actual performance of an architecture, *i.e.*, the proxy score of a set of architecture and the actual performance of them are highly rank correlated [1, 2, 12, 17, 20, 23]. On the other hand, the study in [12] indicates that there is a certain degree of complementarity among several existing zero cost proxies; we therefore combine them through our method. Assuming that we have $\mathcal{M}$ different proxies, the objective is to find the architecture with the highest unified proxy which is weighted sum of the existing proxies,

$$\max_{\mathcal{A}} \text{UP}(A) = \max_{\mathcal{A}} \sum_{i=1}^{\mathcal{M}} \lambda^i \cdot f_{zc}^i(\mathcal{A}). \tag{1}$$

3.2. Preliminary

We adopt the implementation of Variational Graph Isomorphism Autoencoder (VGAE) [11] by [40] to transform the architecture search space into a continuous embedding space. Our search space is cell-based with each cell expressed by a labeled Directed Acyclic Graph (DAG) denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ represents a set of N vertices, and $\mathcal{E}$ represents a set of labeled edges. Each edge is linked to a label selected from a set of K predefined operations. We use an upper triangular adjacency matrix $\mathbf{A} \in \mathbf{R}^{N \times N}$ and an one-hot operation matrix $\mathbf{X} \in \mathbf{R}^{N \times K}$ to encode cell-based neural architectures.

The adjacency matrix is augmented by adding its transpose, denoted as $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{A}^T$. The architecture encoder and decoder used in our approach are depicted below, where the former converts an input architecture to a feature embedding $\mathcal{Z} \in \mathbf{R}^{(N \times F) \times 1}$, F is the dimension of each node embedding, and the latter maps the embedding back to an architecture.

Architecture Encoder. Given the adjacency matrix $\tilde{\mathbf{A}}$ and the one-hot encoded operation matrix $\mathbf{X}$, the encoder model is defined as: $Enc(\tilde{\mathbf{A}}, \mathbf{X}) = \mathcal{N}(\mathbf{Z}|\mu, \sigma^2) = \prod_{i=1}^{N} \mathcal{N}(z_i|\mu_i, diag(\sigma_i^2))$, where we realize the $Enc(\cdot, \cdot)$ using the same setting as Yan et al. [40] to exploit an L-layer Graph Isomorphism Networks (GINs) [38]. The mean and the standard deviation of the encoder are computed as $\mu = GIN_{\mu}(\tilde{\mathbf{A}}, \mathbf{X}), \sigma = GIN_{\sigma}(\tilde{\mathbf{A}}, \mathbf{X})$. z_i is a stochastic latent vector with a dimension of F, and $\mathbf{Z}$ is an $N \times F$ matrix summarizing z_i. In the inference stage, we flatten μ to obtain the vector $\mathcal{Z}$ as the training data of our Multi-Proxy Estimator.

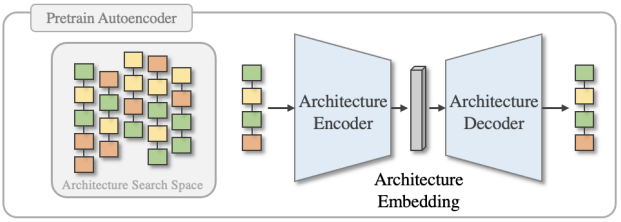


Figure 3. Architecture Autoencoder. We adopt a pretrained architecture autoencoder from VGAE [11] to transform architecture into a continuous space.

Architecture Decoder. Given the latent matrix $\mathbf{Z}$, the decoder aims to reconstruct $\hat{\mathbf{A}}$ and $\hat{\mathbf{X}}$, and the decoder model is defined as follows: $Dec(\mathbf{Z}) = p(\hat{\mathbf{A}}, \hat{\mathbf{X}}|\mathbf{Z}) = p(\hat{\mathbf{A}}|\mathbf{Z})p(\hat{\mathbf{X}}|\mathbf{Z})$, where $p(\cdot)$ denotes a probability density function, and the reconstructed $\hat{\mathbf{A}}$ can be sampled from: $p(\hat{\mathbf{A}}|\mathbf{Z}) = \prod_{i=1}^{N} \prod_{j=1}^{N} p(\tilde{\mathbf{A}}_{ij}|z_i, z_j)$, with $p(\hat{\mathbf{A}}_{ij} = 1|z_i, z_j) = \omega(z_i^T z_j)$, where $\omega(\cdot)$ is the sigmoid activation. Similarly, the reconstructed $\hat{\mathbf{X}}$ can be sampled from $p(\hat{\mathbf{X}} = [k_1, ..., k_N]^T|\mathbf{Z}) = \prod_{i=1}^{N} \text{softmax}(\mathbf{W}_o\mathbf{Z} + \mathbf{b}_o)_{i,k_i}$, where softmax$(\cdot)$ is the softmax activation applied row-wise, $k_n \in \{1, 2, ..., K\}$ indicates the operation selected from the predefined set of K operations at the n^{th} node, $\mathbf{W}_o$ and $\mathbf{b}_o$ are learnable weights and biases of the decoder, respectively.

Training Objective of Autoencoder. In pretraining stage as shown in Figure 3, the Autoencoder are optimized by maximizing the variational lower bound $\mathcal{L}$:

$$\begin{aligned} \mathcal{L} = \mathbb{E}_{\mathcal{N}(\mathbf{Z}|\mu,\sigma^2)}[\log p(\hat{\mathbf{X}}, \hat{\mathbf{A}}|\mathbf{Z})] - \\ \mathcal{D}_{KL}(\mathcal{N}(\mathbf{Z}|\mu, \sigma^2)||p(\mathbf{Z})], \end{aligned} \tag{2}$$

where $p(\hat{\mathbf{X}}, \hat{\mathbf{A}}|\mathbf{Z}) = p(\hat{\mathbf{A}}|\mathbf{Z})p(\hat{\mathbf{X}}|\mathbf{Z})$, $\mathcal{D}_{KL}$ is the Kullback-Leibler divergence.

3.3. Multi-Proxy Estimator (MPE)

The architecture embedding has the advantage of providing a continuous representation of the architecture space. It facilitates dense and continuous architecture search, and the embedding can be converted back to the corresponding architecture easily through the decoder.

In our training phase (as depicted in Figure 2), we train Multi-Proxy Estimator that maps an architecture to the proxy score(s), defined as $f_{MPE}(\mathcal{A}; \mathcal{W}) \in \mathbf{R}^{\mathcal{M}}$, where $\mathcal{A}$ represents an architecture, $\mathcal{W}$ denotes the MLP weights to learn, and $\mathcal{M}$ stands for the number of proxies. Our MPE comprises a pretrained architecture encoder and a

two-hidden-layer perceptron module, each layer equipped with a linear-batchnorm-ReLU triplet. The MLP is employed to simultaneously predict the $\mathcal{M}$ proxy scores from the architecture embedding. The training objective of MPE is to minimize the mean squared error loss as follows:

$$\mathcal{L}_{MPE}(\mathcal{W}) = s^\top s, \tag{3}$$

where $s \in \mathbf{R}^{\mathcal{M}}$ is the difference vector between the predicted and proxy scores,

$$s = (f_{MPE}(\mathcal{A}; \mathcal{W}) - [f^1_{zc}(\mathcal{A}), f^2_{zc}(\mathcal{A}), ...f^{\mathcal{M}}_{zc}(\mathcal{A})]), \tag{4}$$

where $f^i_{zc}(\cdot)$ is the i-th zero-cost proxy for $i = 1, ..., \mathcal{M}$.

3.4. Gradient Ascent

Recall that our goal is to search for an architecture that maximizes the objective function $\sum_{i=1}^{\mathcal{M}} \lambda^i f^i_{zc}(\mathcal{A})$. We apply Tree-structured Parzen Estimator Approach (TPE) [4] to search for optimal proxy weights $\lambda^i, \forall i$ that maximize an order-preserving loss (the rank correlation Kendall's τ) only on a small subset of benchmark data. The optimized proxy weights are then fixed and used to perform the gradient ascent search on other spaces/datasets. In other words, we utilize the proxy weights in a space- and dataset-agnostic manner. See the paragraph of proxy weights in Section 3.5 for more details.

In our searching stage (as illustrated in Figure 2), we iteratively optimize the architecture embedding vector via the frozen MPE (including the architecture encoder, MLP weights $\mathcal{W}$, and combination weights $\Lambda = [\lambda_1, \lambda_2 ... \lambda_M]$) and apply gradient ascent to maximize the objective below.

$$F(\mathcal{Z}) = \sum_{i=1}^{\mathcal{M}} \lambda^i f_{MLP}(\mathcal{Z}; \mathcal{W})_i, \tag{5}$$

which is trained by the ascent learning rule,

$$\mathcal{Z}^{t+1} = \mathcal{Z}^t + \eta \frac{\partial}{\partial \mathcal{Z}} F(\mathcal{Z}), \text{ for } t = 1 \text{ to } T, \tag{6}$$

with η the learning rate and T the number of iterations. Finally, the obtained $\mathcal{Z}^T$ is then decoded back to the architecture space, using the pretrained Architecture Decoder shown in Figure 3.

3.5. Implementations

In this section, we present several implementation details of the proposed approach.

Zero-cost Proxies. To get the zero cost proxy scores, we use the NASLib [26] implementation for NAS-Bench-201 [12]. Both Nas-Bench-201 and DARTS search spaces will be detailed in Section 4. Since Krishnakumar *et al*. [12] have provided proxy scores for about 11k architectures sampled in DARTS space [18], we adopt them directly.

Table 1. The combined weights of each zero-cost proxies we used when performing gradient ascent in different settings.

Proxy	UP$_{sum}$	UP$_{sum-}$	UP$_{weighted}$	UP$_{selected}$
Snip [14]	1	1	-0.164	0
Flops [24]	1	1	0.609	1
Params [24]	1	1	-0.587	0
l2_norm [1]	1	1	-0.100	0
Grasp [33]	1	0	0.441	0
grad_norm [1]	1	1	0.089	0
Synflow [30]	1	1	0.840	1
jacov [23]	1	1	1.000	1
EPE_NAS [20]	1	1	0.094	0
Zen [17]	1	1	0.416	0
fisher [32]	1	1	-0.342	0
plain [1]	1	1	0.275	0
Nwot [23]	1	1	0.138	1

The zero cost proxies we use contain *snip, flops, params, l2-norm, Grasp, grad-norm, synflow, jacov, epe-nas, zen score, fisher, plain, Nwot*. For a single architecture in DARTS space, *grasp* takes about 10 seconds to compute, whereas each of the other 12 proxies takes about 1 second.

Graph Embedding of Architecture. We adopt the implementation of Variational Graph Isomorphism Autoencoder [40] and use the provided model weights to encode the cell structure of both NAS-Bench-201 and DARTS spaces. The embedding dimension is set to 128 and 352, respectively.

Proxy Weights. Finding a good way to combine existing proxies is an open problem. A naive approach is to simply add proxy scores altogether with an equal weight (called 'Sum' in our tables). We refer solving the weight λ^i for each zero-cost proxy f^i_{zc} as a hyperparameter search problem, and we only use a small subset (the Cifar-10 subset of NAS-Bench-201) for finding the weights to ensure their generalibility. We apply Tree-structured Parzen Estimator Approach (TPE), implemented by [3], to search for weights that maximize the rank correlation Kendall's τ between NAS-Bench-201-CIFAR-10's validation accuracy. We show 4 different proxy weights settings including *sum*, *sum-*, *weighted* and *selected*. The *sum* and *sum-* assign equal weights for all of their components. For *selected*, we simply round the weights from searched weights other than *weighted*, but use the same search procedure. Note that we exclude *grasp* from *sum-* due to its excessively long computation time. The searched weights are shown in Table 1.

Multi-Proxy Estimator. We only require zero-cost proxy scores to train our MPE, not the precise accuracy of the model architecture.

For the NAS-Bench-201 space, we take only 50% of the architectures together with their proxy scores as the data for training the MPE using Adam optimizer with a learning rate 1×10^{-3}, batch size 16, without weight decay for 70 epochs.

For DARTS space, we generate 10,000 samples for train-

Table 2. Rank Correlations Kendall's τ of original proxies and predicted proxies on NAS-Bench-201-CIFAR-10.

Proxy	Original score's τ value	Predicted score's τ value
Snip [14]	0.408	0.414
Flops [24]	0.500	0.491
Params [24]	0.538	0.516
l2_norm [1]	0.489	0.478
Grasp [33]	0.214	0.217
grad_norm [1]	0.403	0.401
Synflow [30]	0.537	0.540
jacov [23]	0.520	0.461
EPE_NAS [20]	0.507	0.539
Zen [17]	0.240	0.237
fisher [32]	0.371	0.374
plain [1]	-0.182	-0.369
Nwot [23]	0.574	0.568
UP_{sum}	0.58	0.50
$\text{UP}_{weighted}$	**0.71**	**0.66**
$\text{UP}_{selected}$	0.68	0.59

Table 3. Performance on the test sets of several benchmarks compared to state-of-the-arts on NAS-Bench-201. The best results are in **bold**, and the second best are underlined.

Searched Method	Method	CIFAR-10	CIFAR-100	ImageNet-16-120
Proxy-based	Snip [14]	83.82	49.81	0.83
	Synflow [30]	93.76	71.11	41.44
	Grasp [33]	83.82	49.81	0.83
	EPE_NAS [20]	89.08	62.60	32.30
	Zen [17]	90.65	68.10	40.77
	Nwot [23]	93.32	71.74	46.53
Training-Free	TE-NAS [6]	93.90	71.24	42.38
	Zero-Cost-PT [36]	94.03	**72.53**	46.18
	ξ-GSNR [29]	94.05	72.18	46.24
Ours	UP_{sum}	**94.18**	71.59	**47.16**
	UP_{sum-}	**94.18**	71.59	**47.16**
	$\text{UP}_{weighted}$	**94.18**	71.59	**47.16**
	$\text{UP}_{selected}$	**94.18**	71.59	**47.16**
Upper Bound	-	94.37	73.51	47.31

ing the MPE with Adam optimizer with a learning rate of 5×10^{-4}, batch size 64, without weight decay for 50 epochs.

The training times of MPE on both NAS-Bench-201 and DARTS space take less than 2 minutes.

Gradient Ascent. We randomly sample an architecture from the search space, and then encode it to the embedding space as the initial embedding. Via the frozen MPE trained in the previous step, we optimize the embedding with Adam optimizer with a learning rate of 1×10^{-3}, without weight decay, until the decoded architecture is invalid.

4. Experiments

In this section, we first present the search space. We then show the performance of UP-NAS evaluated on several benchmarks. Finally, we conduct ablation studies.

NAS-Bench-201 search space. NAS-Bench-201 is a well-known search space for NAS, comprising 15,625 cell-based architectures. Each architecture is represented as a directed acyclic graph (DAG), consisting of four nodes and five operations: zero, skip connection, 1×1 convolution, 3×3 convolution, and 3×3 average pooling. To facilitate the NAS research, performance for each architecture is provided for three datasets: NAS-Bench-201-CIFAR-10, NAS-Bench-201-CIFAR-100, and NAS-Bench-201-ImageNet-16-120 (a down-sampled ImageNet dataset containing only 16×16 images of 120 classes).

Table 4. Performance on the test sets of several benchmarks compared to state-of-the-arts on NAS-Bench-201. The best results are in **bold**, and the second best are underlined.

Searched Method	Method	CIFAR-10	CIFAR-100	ImageNet-16-120
Gradient-based	DARTS [18]	54.30	38.97	18.41
	PC-DARTS [39]	93.41	67.48	41.31
	β-DARTS [41]	94.36	**73.51**	46.34
	Shapley-NAS [37]	**94.37**	**73.51**	46.85
Multi-proxy-based	ProxyBO [28]	91.46	73.48	**47.18**
	DELE [43]	**94.37**	73.50	46.39
Ours	UP_{sum}	94.18	71.59	47.16
	UP_{sum-}	94.18	71.59	47.16
	$\text{UP}_{weighted}$	94.18	71.59	47.16
	$\text{UP}_{selected}$	94.18	71.59	47.16
Upper Bound	-	94.37	73.51	47.31

All architectures in ths benchmark have been provided with the associated accuracy values for the three subsets, enabling studing NAS algorithms without model evaluation.

DARTS search space. DARTS space is another widely used search space. It is built by a cell-based representation for neural networks, consisting of a normal cell and a reduction cell, each containing seven nodes. The first two nodes are outputs from two previous cells and the last node depth-wisely concatenates all of the four intermediate nodes. The DAG is created by connecting the remaining four intermediate nodes where each of them is connected by two previous nodes, and the final network is obtained by stacking the cells. The DARTS operation space comprises eight options: none (zero), skip connection, separable convolution 3×3 and 5×5, dilated separable convolution 3×3 and 5×5, max pooling 3×3, and average pooling 3×3. For a fair comparison, we follow the training procedure of [37].

4.1. Rank Correlation on NAS-Bench-201

Since the ground-truth accuracy has been provided for NAS-Bench-201, we can compute the rank correlation between the true accuracy and the proxy scores as an essential evaluation. In Table 2, we show the rank correlations of the original zero-cost proxy scores (middle column) and that of the predicted score obtained using our MPE (third column). It reveals that the scores predicted by MPE can provide roughly consistent rank correlations of the original zero-cost proxies. In addition, the weighted aggregation setting and the associated unified proxy predicted by our MPE

Table 5. Performance on ImageNet dataset with comparisons to state-of-the-arts on the DARTS search space.

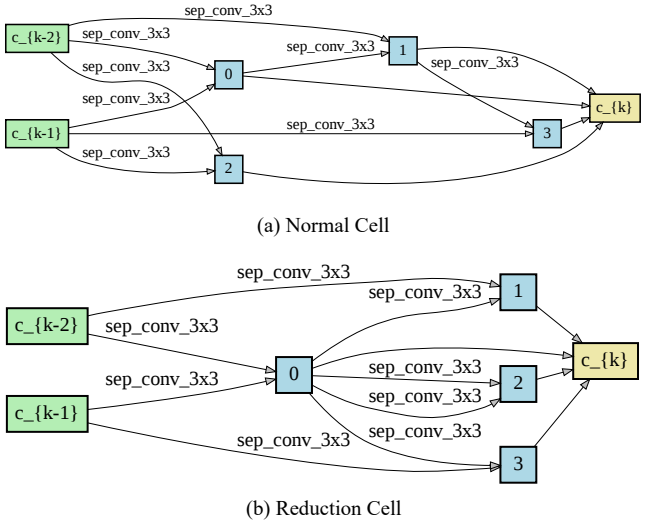
Searched Method	Method	Top-1 Error(%)	Params (M)	Search Cost (GPU days)	Search dataset
Gradient-based	DARTS(2nd order) [18]	26.7	4.7	4.0	CIFAR-10
	β-DARTS [41]	23.9	5.5	0.4	CIFAR-10
	PC-DARTS [39]	25.1	5.3	0.1	CIFAR-10
	PC-DARTS [39]	24.2	5.3	3.8	ImageNet
	Shapley-NAS [37]	24.3	5.1	0.3	CIFAR-10
	Shapley-NAS [37]	23.9	5.4	4.2	ImageNet
Predictor-based	NAONet [21]	25.7	11.4	200	ENAS
	DELE [43]	24.4	4.1	300 queries	CIFAR-10
Training-free	TE-NAS [6]	24.5	5.4	0.17	ImageNet
	Zero-Cost-PT [36]	24.4	6.3	0.018	CIFAR-10
	ξ-GSNR [29]	24.5	5.5	0.017	CIFAR-10
Extensible Proxy	Eproxy [16]	25.7	4.9	0.02	CIFAR-10
	Eproxy+DPS [16]	24.4	5.3	0.06	CIFAR-10
Ours	UP$_{weighted}$	24.5	5.7	**5 sec**	CIFAR-10
	UP$_{selected}$	**23.5**	6.5	**5 sec**	CIFAR-10

can achieve the highest rank correlations, showing their better ordinal-fitting capabilities to the true accuracy than the individual zero-cost proxies.

4.2. Evaluation Results on NAS-Bench-201 Space

In this session, we deomnstrate the performance of our approach on searching the architectures in the NAS-Bench-201 search space. Since our UP-NAS requires no training of the architecture weights, we compare it with the state-of-the-art zero-cost-proxy and training-free approaches on NAS-Bench-201 at first. The performance on the three subsets in the benchmark is shown in Table 3. As can be seen, all settings of our method find the same architecture in this search space. Our method achieves the best test accuracy on most subesets (CIFAR-10 and ImageNet-16-120) among all approaches, while maintaining comparable performance to the leading zero-cost proxy on the CIFAR-100 subset. The last row shows the upper bounds of this benchmark (w.r.t. the architecture of the highest accuracy in the NAS-Bench-201 search space), and our approach can also achieve very close accuracy to the bounds of the CIFAR-10 and ImageNet-16-120 subsets. The results reveal that our UP-NAS can leverage existing zero-cost proxies and unify them for a better score-based search. It can search smoothly in a continuous architecture-embedding space for finding more favorable solutions.

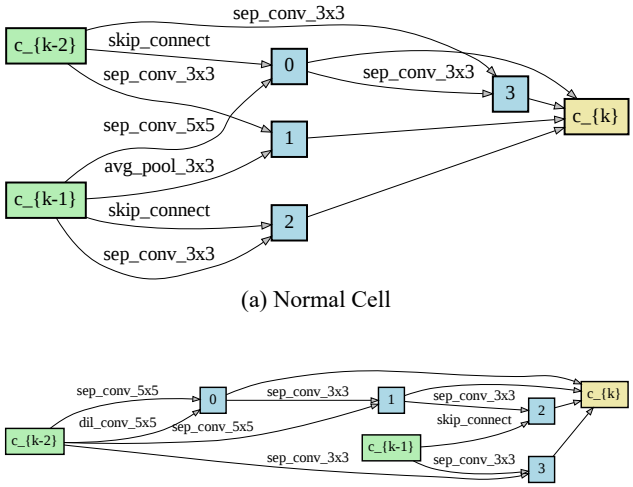
We also compare our UP-NAS with the methods achieving state-of-the-art performance on this benchmark while requiring the model weights training, including gradient-based methods [37, 39, 41], and the methods that utilize multiple zero-cost proxies to improve (but not training-free): ProxyBO [28] and DELE [43]. As shown in Table 4, our UP-NAS still has very competitive performance compared to the non-training-free approaches.



(a) Normal Cell

(b) Reduction Cell

Figure 4. The searched architecture on DARTS-CIFAR-10 using UP-selected. (a) the normal cell (b) the reduction cell. This architecture achieves 23.5% error on ImageNet.



(a) Normal Cell

(b) Reduction Cell

Figure 5. The searched architecture on DARTS-CIFAR-10 using UP-weighted. (a) the normal cell (b) the reduction cell. This architecture achieves 24.5% error on ImageNet.

4.3. Evaluation on DARTS Space for ImageNet

The DARTS space is much larger, where the true accuracies have not been provided for the architectures in the space. To further verify the effectiveness of our approach, we search the architectures in the DARTS space by using UP-NAS and demonstrate the performance on the ImageNet dataset [27]. Consider that several zero-cost proxies we adopt depend not only on the architecture but also on the dataset [17, 20]. To simplify the search, we follow some of the previous studies [16, 18, 39, 41] that search the architectures for a dataset containing fewer and smaller-size images at first (where CIFAR-10 [13] is adopted), and then we apply the searched

Table 6. CIFAR-10 classification error comparison with baseline methods (random search and evolutionary search), searched on the DARTS CNN search space. Note that 1,500 queries for random and evolutionary search methods take around 5 seconds.

Method	Error(%)	Params(M)	Search Cost
Random$_{weighted}$	2.88	3.1	1500 queries $\approx$ 5 sec
Random$_{selected}$	3.22	3.8	1500 queries $\approx$ 5 sec
Evolutionary$_{weighted}$	2.75	4.1	1500 queries $\approx$ 5 sec
Evolutionary$_{selected}$	2.93	4.8	1500 queries $\approx$ 5 sec
Gradient Ascent$_{weighted}$	**2.44**	3.6	5 sec
Gradient Ascent$_{selected}$	2.97	4.7	5 sec

architecture to the large-scaled dataset, ImageNet.

We thus use our UP-NAS that unifies the zero-cost proxies (listed in Table 1) to perform gradient-ascent search in the DARTS space based on the CIFAR-10 dataset at first. We employ the *weighted* and *selection* settings; the architectures found by the associated UP$_{weighted}$ and UP$_{selected}$ are shown in Figure 4 and 5, respectively. The searched architectures (based on CIFAR-10) are then applied to the ImageNet dataset. The results are shown in Table 5.

We compare our UP-NAS with state-of-the-art NAS methods on the ImageNet dataset, which include (1) Gradient-based method DARTS [18] and its recent variants [37, 39, 41], (2) predictor-based methods [21, 43] related to ours, (3) training-free methods [6, 29, 36] and (4) extensible proxy [16]. As shown in Table 5, our method can achieve the most favorable performance on ImageNet in the DARTS search space, which outperforms not only training-free methods but also recent non-training-free methods. Figure 4 shows the best-performed architecture (23.5% error rate) found by our UP-NAS, which can achieve the SOTA performance on ImageNet among the NAS methods in the DARTS search space. The results reveal again that our UP-NAS can well unify the zero-cost proxies and smoothly explore architectures through gradient ascent.

4.4. Ablation Studies

In this section, we show several ablation studies to validate the efficacy of our method.

4.4.1 Search Methods

To verify the effectiveness of the "gradient ascent" method adopted in the proposed approach, we also show the results of two baseline search methods, *random search* and *evolutionary search*. In random search, we randomly sample the architectures in the search space and return the one with the highest predicted weighted proxy sum. In evolutionary search, we keep a population of 100 models, and mutate the candidate models iteratively.

We ablate the experimental study on the CIFAR-10 dataset in the DARTS search space. As shown in Table 6, our approach that employs gradient-ascent-based search achieves the best performance (of an error rate in 2.44%) given the same amount of search cost, where 1,500 queries for random and evolutionary search methods also take around 5 seconds as the search cost. The results demonstrate the effectiveness of gradient-based search.

Table 7. Comparison between different MLP settings on NAS-Bench-201.

Method	CIFAR-10	
	validation	test
MPE$_1$+ UP$_{sum}$	91.18	93.83
MPE$_2$+ UP$_{sum}$	**91.31**	94.18
MPE$_3$+ UP$_{sum}$	91.17	**94.22**

4.4.2 Different MLPs in MPE

We also provide the ablations of determining the architecture of the MLP used in our MPE. We set the number of neurons of each hidden layer as 256, and evaluate the performance of the MLP with the number of hidden layers from 1 to 3 of UP$_{sum}$ setting on NAS-Bench-201. As shown in Table 7, the two-hidden-layer MLP achieves the best validation accuracy on CIFAR-10. Thus, we apply this to our experiments.

5. Conclusions and Future Work

To our knowledge, no single proxy works the best for different tasks and scenarios. We thus propose unified proxy for NAS which is a generalized scoring function. It is realized by an architecture encoder and MLPs learning to predict multiple zero-cost proxies that are unified to consolidate the strength while reducing search bias. We show how to find a new architecture through gradient ascent using the proposed search approach. With extensive experiments, the proposed method achieves competitive performance and significantly reduces the computational resources required for NAS. On the search spaces of DARTS, our approaches achieves the SOTA performance (76.5% top-1 accuracy) on ImageNet. We thus provide a promising solution for training-free NAS.

In the future, we plan to search for vision transformers. There are few proxies (such as [44]) designed specifically for the Transformer. Our approach has the potential to consolidate existing transformer proxies to assist in the architecture search for the new category of architectures.

Acknowledgment. This work was supported in part by the National Science and Technology Council, Taiwan under Grant NSTC 112-2221-E-002-132-MY3 and 112-2634-F-002-005. We thank to National Center for High-performance Computing (NCHC) of National Applied Research Laboratories (NARLabs) in Taiwan for providing computational and storage resources.

References

[1] Mohamed S Abdelfattah, Abhinav Mehrotra, Łukasz Dudziak, and Nicholas Donald Lane. Zero-cost proxies for lightweight nas. In *ICLR*, 2021. 1, 2, 3, 4, 5, 6

[2] Yash Akhauri, J Pablo Munoz, Nilesh Jain, and Ravi Iyer. Evolving zero cost proxies for neural architecture scoring. *arXiv preprint arXiv:2209.07413*, 2022. 2, 4

[3] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *SIGKDD*, 2019. 5

[4] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *NeurIPS*, 2011. 5

[5] Thomas C. P. Chau, Lukasz Dudziak, Mohamed Saleh Abdelfattah, Royson Lee, Hyeji Kim, and Nicholas D. Lane. Brp-nas: Prediction-based nas using gcns. *NeurIPS*, 2020. 2

[6] Wuyang Chen, Xinyu Gong, and Zhangyang Wang. Neural architecture search on imagenet in four gpu hours: A theoretically inspired perspective. In *ICLR*, 2021. 3, 6, 7, 8

[7] Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Bichen Wu, Zijian He, Zhen Wei, Kan Chen, Yuandong Tian, Matthew Yu, Peter Vajda, and Joseph E. Gonzalez. Fbnetv3: Joint architecture-recipe search using predictor pretraining. In *CVPR*, 2021. 2

[8] Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *ICLR*, 2020. 2

[9] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *ECCV*, 2020. 1

[10] Kun Jing, Jungang Xu, and Pengfei Li. Graph masked autoencoder enhanced predictor for neural architecture search. In *IJCAI*, 2022. 2

[11] Thomas Kipf and Max Welling. Variational graph auto-encoders. *ArXiv*, abs/1611.07308, 2016. 4

[12] Arjun Krishnakumar, Colin White, Arber Zela, Renbo Tu, Mahmoud Safari, and Frank Hutter. Nas-bench-suite-zero: Accelerating research on zero cost proxies. *NeurIPS*, 2022. 2, 3, 4, 5

[13] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 7

[14] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. SNIP: SINGLE-SHOT NETWORK PRUNING BASED ON CONNECTION SENSITIVITY. In *ICLR*, 2019. 2, 3, 5, 6

[15] Guihong Li, Duc Hoang, Kartikeya Bhardwaj, Ming Lin, Zhangyang Wang, and Radu Marculescu. Zero-shot neural architecture search: Challenges, solutions, and opportunities. *arXiv preprint arXiv:2307.01998*, 2023. 2

[16] Yuhong Li, Jiajie Li, Cong Hao, Pan Li, Jinjun Xiong, and Deming Chen. Extensible and efficient proxy for neural architecture search. In *ICCV*, pages 6199–6210, 2023. 7, 8

[17] Ming Lin, Pichao Wang, Zhenhong Sun, Hesen Chen, Xiuyu Sun, Qi Qian, Hao Li, and Rong Jin. Zen-nas: A zero-shot nas for high-performance image recognition. In *ICCV*, 2021. 1, 3, 4, 5, 6, 7

[18] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *ICLR*, 2019. 1, 2, 5, 6, 7, 8

[19] Liyang Liu, Shilong Zhang, Zhanghui Kuang, Aojun Zhou, Jing-Hao Xue, Xinjiang Wang, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. Group fisher pruning for practical network compression. In *ICML*, pages 7021–7032. PMLR, 2021. 3

[20] Vasco Lopes, Saeid Alirezazadeh, and Luís A. Alexandre. Epe-nas: Efficient performance estimation without training for neural architecture search. In *ICANN*, 2021. 1, 3, 4, 5, 6, 7

[21] Renqian Luo, Fei Tian, Tao Qin, En-Hong Chen, and Tie-Yan Liu. Neural architecture optimization. In *NeurIPS*, 2018. 2, 7, 8

[22] Renqian Luo, Xu Tan, Rui Wang, Tao Qin, Enhong Chen, and Tie-Yan Liu. Semi-supervised neural architecture search. *NeurIPS*, 2020. 2

[23] Joe Mellor, Jack Turner, Amos Storkey, and Elliot J Crowley. Neural architecture search without training. In *ICML*, 2021. 1, 3, 4, 5, 6

[24] Xuefei Ning, Changcheng Tang, Wenshuo Li, Zixuan Zhou, Shuang Liang, Huazhong Yang, and Yu Wang. Evaluating efficient performance estimators of neural architectures. In *NeurIPS*, 2021. 3, 5, 6

[25] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *ICML*, 2018. 1

[26] Michael Ruchte, Arber Zela, Julien N. Siems, Josif Grabocka, and Frank Hutter. Naslib: A modular and flexible neural architecture search library. 2020. 3, 5

[27] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115:211–252, 2015. 7

[28] Yu Shen, Yang Li, Jian Zheng, Wentao Zhang, Peng Yao, Jixiang Li, Sen Yang, Ji Liu, and Bin Cui. Proxybo: Accelerating neural architecture search via bayesian optimization with zero-cost proxies. In *AAAI*, 2023. 3, 6, 7

[29] Zihao Sun, Yu Sun, Longxing Yang, Shun Lu, Jilin Mei, Wenxiao Zhao, and Yu Hu. Unleashing the power of gradient signal-to-noise ratio for zero-shot nas. In *ICCV*, pages 5763–5773, 2023. 3, 6, 7, 8

[30] Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. In *NeurIPS*, 2020. 2, 3, 5, 6

[31] Yehui Tang, Yunhe Wang, Yixing Xu, Hanting Chen, Boxin Shi, Chao Xu, Chunjing Xu, Qi Tian, and Chang Xu. A semi-supervised assessor of neural architectures. In *CVPR*, 2020. 2

[32] Jack Turner, Elliot J. Crowley, Michael O'Boyle, Amos Storkey, and Gavin Gray. Blockswap: Fisher-guided block substitution for network compression on a budget. In *ICLR*, 2020. 3, 5, 6

[33] Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. In *ICLR*, 2020. 3, 5, 6

[34] Ruochen Wang, Minhao Cheng, Xiangning Chen, Xiaocheng Tang, and Cho-Jui Hsieh. Rethinking architecture selection in differentiable nas. In *ICLR*, 2021. 3

[35] Chen Wei, Chuang Niu, Yiping Tang, and Jimin Liang. Npenas: Neural predictor guided evolution for neural architecture search. *IEEE transactions on neural networks and learning systems*, 2020. 2

[36] Lichuan Xiang, Łukasz Dudziak, Mohamed S Abdelfattah, Thomas Chau, Nicholas D Lane, and Hongkai Wen. Zero-cost proxies meet differentiable architecture search. *AAAI*, 2023. 3, 6, 7, 8

[37] Han Xiao, Ziwei Wang, Zheng Zhu, Jie Zhou, and Jiwen Lu. Shapley-nas: Discovering operation contribution for neural architecture search. In *CVPR*, 2022. 6, 7, 8

[38] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2019. 4

[39] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. Pc-darts: Partial channel connections for memory-efficient architecture search. In *ICLR*, 2020. 6, 7, 8

[40] Shen Yan, Yu Zheng, Wei Ao, Xiao Zeng, and Mi Zhang. Does unsupervised architecture representation learning help neural architecture search? *NeurIPS*, 2020. 2, 4, 5

[41] Peng Ye, Baopu Li, Yikang Li, Tao Chen, Jiayuan Fan, and Wanli Ouyang. b-darts: Beta-decay regularization for differentiable architecture search. In *CVPR*, 2022. 1, 6, 7, 8

[42] Yun Yi, Haokui Zhang, Wenze Hu, Nannan Wang, and Xiaoyu Wang. Nar-former: Neural architecture representation learning towards holistic attributes prediction. In *CVPR*, pages 7715–7724, 2023. 2

[43] Junbo Zhao, Xuefei Ning, Enshu Liu, Binxin Ru, Zixuan Zhou, Tianchen Zhao, Chen Chen, Jiajin Zhang, Qingmin Liao, and Yu Wang. Dynamic ensemble of low-fidelity experts: Mitigating nas "cold-start". In *AAAI*, pages 11316–11326, 2023. 3, 6, 7, 8

[44] Qinqin Zhou, Kekai Sheng, Xiawu Zheng, Ke Li, Xing Sun, Yonghong Tian, Jie Chen, and Rongrong Ji. Training-free transformer architecture search. In *ICCV*, pages 10894–10903, 2022. 8