Zero-Shot Neural Architecture Search: Challenges, Solutions, and Opportunities

Guihong Li, *Student Member, IEEE,* Duc Hoang, *Student Member, IEEE,*
Kartikeya Bhardwaj, *Member, IEEE,* Ming Lin, *Member, IEEE,*
Zhangyang Wang, *Senior Member, IEEE,* Radu Marculescu, *Fellow, IEEE*

Abstract—Recently, *zero-shot* (or *training-free*) Neural Architecture Search (NAS) approaches have been proposed to liberate NAS from the expensive training process. The key idea behind zero-shot NAS approaches is to design proxies that can predict the accuracy of some given networks without training the network parameters. The proxies proposed so far are usually inspired by recent progress in theoretical understanding of deep learning and have shown great potential on several datasets and NAS benchmarks. This paper aims to comprehensively review and compare the state-of-the-art (SOTA) zero-shot NAS approaches, with an emphasis on their hardware awareness. To this end, we first review the mainstream zero-shot proxies and discuss their theoretical underpinnings. We then compare these zero-shot proxies through large-scale experiments and demonstrate their effectiveness in both hardware-aware and hardware-oblivious NAS scenarios. Finally, we point out several promising ideas to design better proxies. Our source code and the list of related papers are available on https://github.com/SLDGroup/survey-zero-shot-nas.

Index Terms—Neural Architecture Search, Zero-shot proxy, Hardware-aware neural network design

1 Introduction

In recent years, deep neural networks have made significant breakthroughs in many applications, such as recommendation systems, image classification, and natural language modeling [1], [2], [3], [4], [5], [6], [7]. To automatically design high performance deep networks, *Neural Architecture Search* (NAS) has been proposed during the past decade [8], [9], [10], [11], [12]. Specifically, NAS boils down to solving an optimization problem with specific targets (*e.g.*, high classification accuracy) over a set of possible candidate architectures (search space) within a group of computational budgets. Recent breakthroughs in NAS simplify the trial-and-error manual architecture design process and discover new deep network architectures with better performance and efficiency over hand-crafted ones [10], [11], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28]. Therefore, NAS has attracted significant attention from both academia and industry.

One important application of NAS is to design hardware efficient deep models under various constraints, such as memory footprint, inference latency, and power consumption [29]. Roughly, existing NAS approaches can be categorized into three groups as shown in Figure 1: multi-shot NAS, one-shot NAS and zero-shot NAS. Multi-shot NAS methods involve training multiple candidate networks and are therefore time-consuming. It can take from a few hundred GPU hours [30] to thousands of GPU hours [31] in multi-shot NAS methods. One-shot NAS methods alleviate the computational burden by sharing candidate operations via a hyper-network [11], [32], [33], [34], [35], [36], [37]. As shown in Figure 2, one-shot NAS only needs to train a single hyper-network instead of multiple candidate architectures whose number is usually exponentially large. The orders of magnitude reduction in training time enables differentiable search to achieve competitive accuracy against multi-shot NAS, but with much lower search costs [11].

Nevertheless, naively merging all candidate operations into a hyper-network is not efficient because the parameters of all operations need to be stored and updated during the search process. Consequently, the *weight-sharing* methods improve the search efficiency of NAS even further [13], [39], [40], [41], [42]. As shown in Figure 3, the key idea of weight-sharing NAS is to share the parameters across different operations. Next, at each training step, a sub-network is sampled from the hyper-network and then the updated parameters are copied back to the hyper-network. By sharing the parameters of various sub-networks, this differentiable search approach significantly reduces the search costs to a few or tens of GPU hours [39].

Though the differentiable search and weight-sharing have significantly improved the time efficiency of NAS, training is still required in one-shot NAS methods. In the last few years, the *zero-shot* NAS has been proposed to liberate NAS from parameter training entirely [43], [44], [45], [46], [47], [48], [49], [50], [51], [52].

Compared to multi-shot and one-shot methods, zero-shot NAS has the following major advantages: (*i*) **Time efficiency**: zero-shot NAS utilizes some proxy as the model's test accuracy to eliminate the model training altogether during the search stage. Compared to model training, the computation costs of these proxies are much more lightweight. Therefore, zero-shot NAS can significantly reduce the costs of NAS while achieving comparable test accuracy as one-

- *Guihong Li, Duc Hoang, Zhangyang Wang, and Radu Marculescu are with the Department of Electrical and Computer Engineering, The University of Texas at Austin, TX, 78712. E-mail: {lgh, hoangduc, atlaswang, radum}@utexas.edu*
- *Kartikeya Bhardwaj is with Qualcomm AI Research, an initiative of Qualcomm Technologies, Inc., CA, 92121. E-mail: kbhardwa@qti.qualcomm.com*
- *Ming Lin is with Amazon, WA, 98004. E-mail: minglamz@amazon.com.*
- *Correspondence to Radu Marculescu (radum@utexas.edu).*

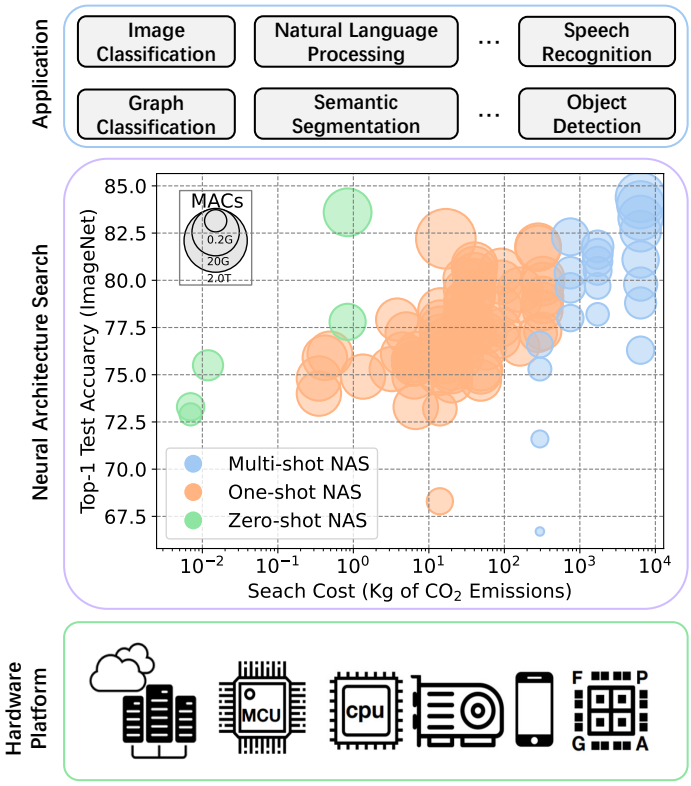


Fig. 1: Overview of existing NAS approaches. NAS is designed to search for optimal architectures with both good accuracy and high efficiency on real hardware. (Data collected from [38])

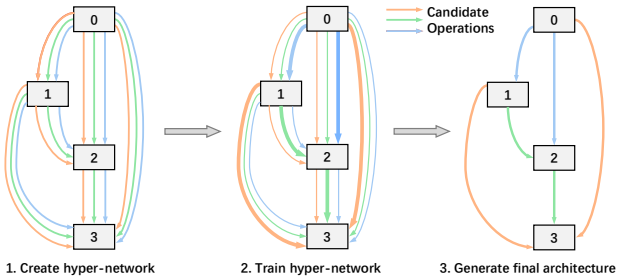


Fig. 2: Illustration of differentiable neural architecture search. (1). Merge all candidate operations into a hyper-network with learnable weights for each operation. (2). Train the hyper-network and update the learnable weights for each operation. (3) Generate the final results by selecting the operations with the highest weight values (boldest edges). (Adapted from [11])

shot and multi-shot NAS approaches (see Figure 1). (*ii*) **Interpretability**: Clearly, the quality of the accuracy proxy ultimately determines the performance of zero-shot NAS. The design of an accuracy proxy for zero-shot NAS is usually inspired by some theoretical analysis of deep neural networks thus deepening the theoretical understanding of why certain networks may work better. For example, Bhardwaj et al. developed the first zero-shot NAS approach by analyzing the topological properties of deep networks [53]; some recent approaches use the number of linear regions to approximate the complexity of a deep neural network [54]. Moreover, the

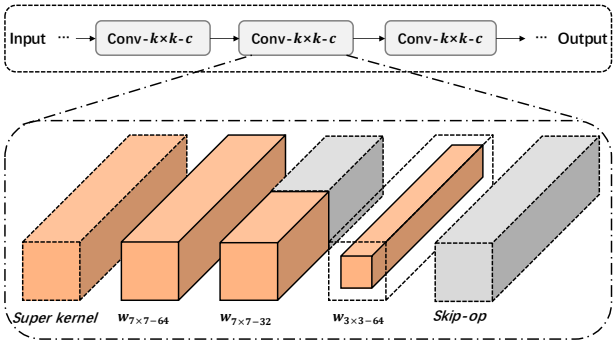


Fig. 3: Illustration of weight-sharing mechanism. The parameters of relatively simple operations are obtained from complex operations, *i.e.*, super kernel. As shown, different operations share the parameters from the super kernel. (Adapted from [39])

connection between the gradient of a network at random initialization and the accuracy of that network after training are widely explored as proxies of the model's test accuracy in zero-shot NAS [55].

Based on these overarching observations, this paper aims to comprehensively analyze existing hardware-aware zero-shot NAS methods. Starting from the theoretical foundations of deep learning, we first investigate various proxies of test accuracy and their theoretical underpinnings. Then, we introduce several popular benchmarks for evaluating zero-shot NAS methods. Moreover, we demonstrate their effectiveness when applied to hardware-aware NAS; notably, we reveal fundamental limitations of existing proxies. Finally, we discuss several potential research directions for hardware-aware zero-shot NAS. Overall, this paper makes the following contributions:

- We review existing proxies for zero-shot NAS and provide theoretical insights behind these proxies. We categorize the existing accuracy proxies into (*i*) gradient-based proxies and (*ii*) gradient-free proxies.
- We conduct direct comparisons of various zero-shot proxies against two naive proxies, *i.e.*, #Params and #FLOPs, and reveal a fundamental limitation of many existing proxies: they correlate much worse with the test accuracy in constrained search settings (*i.e.*, when considering only networks of high accuracies) compared to unconstrained settings (*i.e.*, considering all architectures in the given search space).
- We further conduct a thorough study including proxy design, benchmarks, and real hardware profiling for zero-shot NAS. We show that a few proxies have a better correlation with the test accuracy than these two naive proxies (#Params and #FLOPs) on the top-performing architectures such as ResNets and MobileNets.
- We discuss the limitations of existing zero-shot proxies and NAS benchmarks; we then outline a few possible directions for future research.

In comparison to other existing zero-shot NAS surveys [56], [57], [58], [59], [60], [61], [62], [63], [64], we not only cover all existing proxies, but also provide a deep analysis of

the theoretical underpinning behind them. We believe that understanding the theoretical design considerations behind these proxies is very important for future improvements. Additionally, this is the first work to comprehensively compare these zero-shot proxies on large scale tasks like ImageNet-1K classification, COCO object detection, and ADE20K semantic segmentation. Furthermore, we are the first to explore the potential applicability of these zero-shot proxies to Vision Transformers. Last but not least, we have conducted detailed comparisons for the first time when applying zero-shot NAS in hardware-aware scenarios. This is crucial for deploying the zero-shot approaches in practice, especially for edge-AI applications.

The remaining paper is organized as follows. We introduce zero-shot proxies in Section 2. Section 3 surveys existing NAS benchmarks. Hardware performance predictor is presented in Section 3.2. We evaluate various zero-shot proxies under diverse settings in Section 4 and point out future research directions. We conclude the paper in Section 5.

2 Zero-Shot Proxies

The goal of zero-shot NAS is to design proxies that can rank the accuracy of candidate network architectures at the initialization stage, *i.e.,* without training, such that we can replace the expensive training process in NAS with some computation-efficient alternatives. Hence, the proxy for the accuracy ranking is the key factor of zero-shot NAS.

2.1 Theoretical Underpinning of Proxies

Before we dive deep into the details of existing zero-shot proxies, let us first establish the foundational principles for designing a good zero-shot proxy. Indeed, an ideal accuracy proxy should address three primary aspects [65], [66]:

- **Expressive Capacity**: The proxy should reflect how well the deep network can capture and model complex patterns and relationships within the data, which can be crucial for complex tasks like large-scale datasets (e.g., ImageNet-1K and COCO) [67], [68].
- **Generalization Capacity**: The proxy should also reflect the network ability to generalize from the training data to unseen or out-of-distribution data. A network with a high generalization capacity should not only perform well on the training data but also on new examples, indicating that it has learned meaningful, transferable representations [69], [70], [71].
- **Trainability and Convergence**: The proxy should also indicate how quickly the network converges to a desirable performance level. Faster convergence indicates that the network is efficiently adapting to the training data and task at hand, which is essential for practical applications since training is typically expensive [72], [73], [74].

In short, a good zero-shot proxy for deep network accuracy should provide insights into the network capacity to learn complex representations, generalize to unseen samples, and train to converge to minimal loss values. However, as shown in Table 2, most existing proxies tend to target only one of these aspects. This narrow focus results in outcomes that often fail to outperform some naive proxies, like #Params or #FLOPs; we empirically verify this observation in Section 4.

In this paper, we categorize the existing zero-shot proxies as follows: depending on whether or not the gradients are involved in the proxy calculation, the existing accuracy proxies fall into two major classes: (i) gradient-based accuracy proxy and (ii) gradient-free accuracy proxy (summarized in Table 2). The symbols used in this section and their corresponding meaning are summarized in Table 1.

2.2 Gradient-based accuracy proxies

We first introduce several similar proxies derived from the gradient over parameters of deep networks.

2.2.1 Gradient norm

The gradient norm is the sum of norms for each layer's gradient vector [55]. To calculate the gradient norm, we first input a mini-batch of data into the network and then propagate the loss values backward. Next, we calculate the ℓ_2-norm of each layer's gradient and then add them up for all the convolution and linear layers of the given network. Formally, the definition of gradient norm G is as follows:

$$G \triangleq \sum_{i=1}^{D} \|\nabla_{\boldsymbol{\theta}_i} L\|_2 \tag{1}$$

where D, $\boldsymbol{\theta}_i$ and L are, the number of layers, the parameter vector of the i-th layer of a given network and L is the loss values, respectively.

2.2.2 SNIP

The gradient norm only measures the property of the gradient's propagation for a given network. To jointly measure the parameter importance both in forward inference and gradient propagation, SNIP consists of multiplying the value of each parameter and its corresponding gradient [75]. Formally, SNIP is defined as below:

$$\text{SNIP} \triangleq \sum_{i}^{D} |\langle \boldsymbol{\theta}_i, \nabla_{\boldsymbol{\theta}_i} L \rangle| \tag{2}$$

where $\langle \cdot, \cdot \rangle$ represents the inner product; D, $\boldsymbol{\theta}_i$ and L are, the number of layers, the parameter vector of the i-th layer of a given network and L is the loss values, respectively.

2.2.3 Synflow

Similar to SNIP, Synflow consists of maintaining the sign of the SNIP proxy [76]:

$$\text{Synflow} \triangleq \sum_{i}^{D} \langle \boldsymbol{\theta}_i, \nabla_{\boldsymbol{\theta}_i} L \rangle \tag{3}$$

2.2.4 GraSP

The three proxies mentioned above only take the first-order derivatives of neural networks into account. The GraSP proxy considers both the first-order and second-order derivatives of neural networks [77]. Specifically, GraSP is defined by the inner product of the parameters and the product of the Hessian matrix and the gradients:

$$\sum_{i}^{D} -\langle \boldsymbol{H}_i \nabla_{\boldsymbol{\theta}_i} L, \boldsymbol{\theta}_i \rangle \tag{4}$$

TABLE 1: The symbols used in this paper and their corresponding meaning.

Symbol	Meaning	Symbol	Meaning
$\boldsymbol{x}$	Input samples	$\hat{y}$	Ground truth (labels)
f	A given deep network	D	The number of layers of a given network
f_e	A network w/o final pooling and FC layers	y	The output of a given model
$\mathcal{L}$	Loss function	L	Loss values
$\boldsymbol{\Theta}$	All parameters of a given network	$\boldsymbol{\theta}_i$	Parameters vector of the i-th layer
$\boldsymbol{H}_i$	Hessian matrix of the i-th layer	$\boldsymbol{z}_i$	The output vector of layer i

TABLE 2: Categorization of zero-shot proxies. Based on whether or not the proxy relies on gradients, there are gradient-based and gradient-free approaches. We also categorize existing proxies by their theoretical underpinning (cf. Section 2.1). An empty cell indicates the proxy is not in that category.

Proxy	Grad_norm	SNIP	Synflow	GraSP	GradSign	Fisher	Jacob_cov	NTK_Cond	Zen-score	#LR	Logdet	NN-Mass
Gradient-free										✓	✓	✓
Gradient-based	✓	✓	✓	✓	✓	✓	✓	✓	✓			
Trainability &Convergence	✓	✓	✓	✓	✓		✓	✓				✓
Expressive Capacity						✓			✓	✓	✓	✓
Generalization Capacity								✓				

where $\boldsymbol{H}_i$ is Hessian matrix of the i-th layer.

There are multiple theoretical analyses for the above three proxies. Specifically, Synflow and SNIP have been proven to be layer-wise constants in linear networks during the back-propagation process [75], [76]. Moreover, several works show that Synflow and GraSP are different approximations of the first-order Taylor expansions of deep neural networks [77], [78]. We remark that Taylor expansions of a deep network can identify the parameters that contribute the most to the loss values; thus, it can measure the importance of parameters.

2.2.5 GradSign

Given an input batch with B input samples $\{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_B\}$, GradSign is defined as follows [79]:

$$\text{GradSign} \triangleq \sum_{\theta_k \in \boldsymbol{\Theta}} \left| \sum_{i=1}^{B} \text{sign}[\nabla_{\theta_k} \mathcal{L}\left(f(\boldsymbol{x}_i), y_i\right)] \right| \tag{5}$$

Essentially, GradSign assesses the uniformity across multiple training samples for each parameter, and then adds them up as the final proxy value. It has been proven that GradSign serves as an approximation of the training loss following the training phase [79]. More specifically, a higher value of GradSign is indicative of a diminished training loss. Consequently, GradSign measures the convergence properties inherent in deep neural networks.

Besides the gradient over parameters, the gradient over each layer's activation is also explored to build the accuracy proxy as shown below.

2.2.6 Fisher information

Fisher information of a neural network can be approximated by the square of the activation value and their gradients [80], [81]:

$$\sum_{i}^{D} \langle \nabla_{\boldsymbol{z}_i} L, \boldsymbol{z}_i \rangle^2 \tag{6}$$

where $\boldsymbol{z}_i$ is the feature map vector of the i-th layer of a given network.

Previous works show that a second-order approximation of Taylor expansion in a neural network is equivalent to an empirical estimate of the Fisher information [81]. Hence, measuring the Fisher information of each neuron/channel of a given network can reflect the importance of these neurons/channels.

2.2.7 Jacobian covariant

Besides the gradient over parameters and activations, the Jacobian covariant (Jacob_cov) leverages the gradient over the input data $\boldsymbol{x}$ [82], [83]. To calculate the Jacob_cov proxy, given an input batch with B input samples $\{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_B\}$, the gradients matrix $\boldsymbol{J}$ of the output results $\{y_1, y_2, ..., y_B\}$ w.r.t. these inputs are first computed:

$$\boldsymbol{J} = (\nabla_{\boldsymbol{x}_1} y_1, \nabla_{\boldsymbol{x}_2} y_2, ..., \nabla_{\boldsymbol{x}_B} y_B)^T \tag{7}$$

Next, the raw covariance matrix is generated as:

$$\boldsymbol{G} = (\boldsymbol{J} - \boldsymbol{M})(\boldsymbol{J} - \boldsymbol{M})^T \tag{8}$$

where $M_{i,j} = \frac{1}{B} \sum_{n=1}^{B} J_{i,n}$. Then the raw covariance matrix is normalized to get the real covariance matrix $\boldsymbol{\Gamma}$:

$$\boldsymbol{\Gamma}_{i,j} = \frac{G_{i,j}}{\sqrt{G_{i,i} G_{j,j}}} \tag{9}$$

where $\Gamma_{i,j}$ denotes the entries of $\mathbf{\Gamma}$. Let $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_B$ be the B eigenvalues of $\mathbf{\Gamma}$; then the Jacobian covariant is generated as follows:

$$\text{Jacob\_cov} \triangleq -\sum_{i=1}^{B} \left[(\lambda_i + \epsilon) + (\lambda_i + \epsilon)^{-1} \right] \tag{10}$$

where ϵ is a small value used for numerical stability. As discussed in [82], [83], Jacob_cov can reflect the expressivity of deep networks thus higher Jacob_cov values indicate better accuracy.

2.2.8 Zen-score

Zen-score is a new proxy for a given model [84], [85]. The Zen-score is defined as:

$$\log \mathbb{E}_{\boldsymbol{x},\boldsymbol{\epsilon}} \left(\|f_e(\boldsymbol{n}) - f_e(\boldsymbol{n} + \alpha\boldsymbol{\epsilon})\|_F \right) + \sum_{k,i} \log \left(\sqrt{\frac{\sum_j \sigma_{ij}^k}{Ch_i}} \right), \quad \boldsymbol{x} \sim \mathcal{N}(0, \boldsymbol{I}) \tag{11}$$

where, $\boldsymbol{n}$ is a sampled Gaussian random vector, $\boldsymbol{\epsilon}$ is a small input perturbation, $\|\cdot\|_F$ indicates the Frobenius norm, α is a tunable hyper-parameter, Ch_i is the number of channels of the the i-th convolution layer, and σ_{ij}^k is the variance of the i-th layer's j-th channels for the k-th samples in an input batch data. As shown in Eq.11, Zen-score measures model expressivity by averaging the Gaussian complexity under randomly sampled x and ϵ. We note that this is equivalent to computing the expected gradient norm of f with respect to input x instead of network parameters. Hence, Zen-score measures the expressivity of neural networks instead of their trainability: networks with a higher Zen-score have a better expressivity and thus tend to have a better accuracy.

2.2.9 NTK Condition Number

Neural Tangent Kernel is proposed to study the training dynamics of neural networks [86]. More precisely, given two input samples $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, NTK is defined as:

$$\kappa(\boldsymbol{x}_1, \boldsymbol{x}_2) = \boldsymbol{J}(\boldsymbol{x}_1)\boldsymbol{J}(\boldsymbol{x}_2) \tag{12}$$

where $\boldsymbol{J}(\boldsymbol{x})$ is the Jacobian matrix evaluated at the sample $\boldsymbol{x}$ [87]. Lee et al. prove that the training dynamics of wide neural networks can be solved as follows [88]:

$$\mu_t(\boldsymbol{X}) = \left(\boldsymbol{I} - e^{-\eta t \mathcal{K}(\boldsymbol{X},\boldsymbol{X})} \right) \boldsymbol{y} \tag{13}$$

where t denotes the training step; μ_t represents the output expectations at training step t; $\boldsymbol{X} \in \mathbb{R}^{m \times d}$ and $\boldsymbol{y} \in \mathbb{R}^m$ are the training input having m samples with d dimensions per sample, and their corresponding labels, respectively; η is the learning rate. $\mathcal{K}(\boldsymbol{X}, \boldsymbol{X}) \in \mathbb{R}^{m \times m}$ is the NTK for these input data. By conducting the eigendecomposition of Eq. 13, the i-th dimension in the eigenspace of output expectation can be written as follows:

$$\mu_t(\boldsymbol{X}_i) = \left(\boldsymbol{I} - e^{-\eta \lambda_i t} \right) \boldsymbol{y}_i, i = \{1, 2, ..., m\} \tag{14}$$

where $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_m$ are the eigenvalues of the NTK $\mathcal{K}(\boldsymbol{X}, \boldsymbol{X})$.

Therefore, a smaller difference between λ_1 and λ_m indicates (on average) a more "balanced" convergence among different dimensions in the eigenspace. To quantify the above observation, the NTK Condition Number (NTK_Cond) is defined as follows [54]:

$$\text{NTK\_Cond} \triangleq \mathbb{E}_{\boldsymbol{X},\boldsymbol{\Theta}} \frac{\lambda_m}{\lambda_1} \tag{15}$$

where $\boldsymbol{\Theta}$ is the randomly initialized network parameters. Chen et al. demonstrate that the NTK_Cond is negatively correlated with the architecture's test accuracy [54]. Hence, the networks with lower NTK_Cond values tend to have a higher test accuracy. Similar insights are reported and leveraged in [89] for NAS of vision transformers (ViTs).

2.3 Gradient-free accuracy proxy

Though the gradient-based proxies do not require the training process on the entire dataset, backward propagation is still necessary to compute the gradient. To entirely remove the gradient computation from the neural architecture search, several gradient-free proxies have been proposed lately.

2.3.1 Number of linear regions

The number of linear regions in a neural network indicates the distinct sections into which the network can partition its input space; thus, it describes the expressivity of a given network [90], [91], [92], [93]. For instance, a single-neuron perceptron with a ReLU activation function can divide its input space into two regions. Previous work shows that one can estimate the number of linear regions with the help of the activation patterns in the output activation matrix $\boldsymbol{R}$ [92]:

$$\boldsymbol{R} = \mathbf{1} \cdot \mathbf{1}^T - \text{sign}[\boldsymbol{z}_i(\mathbf{1} - \boldsymbol{z}_i)^T + (\mathbf{1} - \boldsymbol{z}_i)\boldsymbol{z}_i^T] \tag{16}$$

where $\mathbf{1}$ is an all-one vector. Next, by removing the repeating patterns and assigning the weights to each pattern, the number of linear regions ρ is as follows:

$$\rho \triangleq \sum_j \frac{1}{\sum_k R_{j,k}} \tag{17}$$

where $R_{j,k}$ is the entry of R. Therefore, the number of linear regions measures how many unique regions the network can divide the entire activation space into (see Figure 4).

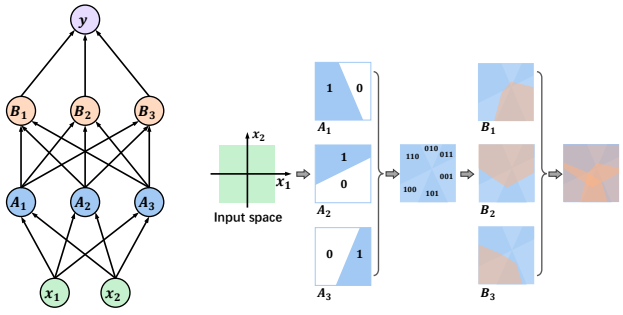


Fig. 4: The illustration of Logdet proxy; $A_i, B_i, i = \{1, 2, 3\}$ are the neurons of a multi-layer perceptron. First, the input space is divided into several linear regions. Next, each region is encoded by a binary code; then Eq. 18 is applied to compute the Logdet proxy. (Adapted from [83])

2.3.2 Logdet

Logdet is another proxy proposed based on the number of linear regions [83]:

$$\boldsymbol{H} = \begin{bmatrix} N_{LR} - d_H(\boldsymbol{c}_1, \boldsymbol{c}_1) & \cdots & N_{LR} - d_H(\boldsymbol{c}_1, \boldsymbol{c}_N) \\ \vdots & & \vdots \\ N_{LR} - d_H(\boldsymbol{c}_N, \boldsymbol{c}_1) & \cdots & N_{LR} - d_H(\boldsymbol{c}_N, \boldsymbol{c}_N) \end{bmatrix}$$
$$\text{Logdet} \triangleq \log|\boldsymbol{H}| \tag{18}$$

where N_{LR} is the total number of linear regions, d_H is the Hamming distance, and $\boldsymbol{c}_i$ is the binary coding vector of the i-th linear region as shown in Figure 4. Previous work shows that networks with a higher Logdet at initialization tend to have higher test accuracy after training [83].

2.3.3 Topology inspired proxies

The very first pioneering work behind theoretically-grounded, training-free architecture design was done by Bhardwaj et al. [53]. While the above proxies are proposed for a general search space, *i.e.,* without any constraints on the candidate architectures, as discussed later, these general-purpose proxies are not better than some naive proxies, *e.g.,* the number of parameters (#Params) of a model. To design better accuracy proxies than #Params, Bhardwaj et al. [53] constrained the search space to specific topologies, e.g., DenseNets, ResNets, MobileNets, etc., and theoretically studied how network topology influences gradient propagation. Inspired by the network science, NN-Mass is defined as follows [53]:

$$\rho_c \triangleq \frac{\#\text{Actual skip connections of cell } c}{\#\text{Total possible skip connections of cell } c}$$
$$\text{NN-Mass} \triangleq \sum_{\text{each cell } c} \rho_c w_c d_c \tag{19}$$

where w_c and d_c are the width and depth values of a cell[1], respectively. Bhardwaj et al. prove that higher NN-Mass values indicate better trainability of networks and faster convergence rate during training [90]. Moreover, they also show that networks with higher NN-Mass values tend to achieve a higher accuracy. NN-Mass has also been used to perform training-free model scaling to significantly improve accuracy-MACs tradeoffs compared to highly accurate models like ConvNexts [94]. In [94], Bhardwaj et al. show the connection between NN-Mass and expressive power of deep networks for ResNet-type networks.

As an extension of NN-Mass, NN-Degree is proposed by relaxing the constraints on the width of networks. Formally, NN-Degree is defined as follows [95]:

$$\text{NN-Degree} = \sum_{\text{each cell } c} (w_c + \frac{\#\text{Actual skip connections}}{\#\text{Total input channels}}) \tag{20}$$

where w_c is the average width value of a cell c. Similarly to NN-Mass, NN-Degree has shown a high positive correlation with the test accuracy.

Lately, Chen et al. developed another principled approach for understanding of a neural network connectivity patterns based on its capacity or trainability [96]. Specifically, they theoretically characterized the impact of connectivity patterns on the convergence of deep networks under gradient descent training with fine granularity, by assuming a wide network and analyzing its Neural Network Gaussian Process (NNGP) [97]. Chen et al. also prove that how the spectrum of an NNGP kernel propagates through a particular connectivity pattern would affect the bounds of the convergence rates. On the practical side, they show that such NNGP-based characterization could act as a simple filtration of "unpromising" connectivity patterns, to significantly accelerate the large-scale neural architecture search without any overhead.

2.4 Summary

As shown in Table 2, most of the existing zero-shot proxies are gradient-based. We note that to calculate the gradient typically involves the backward propagation. Hence, gradient-based proxies are less efficient than gradient-free proxies. Besides, most of the gradient-based proxies (except for Fisher and Logdet), are designed to measure the trainability of deep networks. In contrast, most of the gradient-free proxies (except for NN-Mass) are indicatives of the expressive capacity of neural networks. Moreover, apart from NTK_Cond, current proxies fail to quantify the generalization capacity of deep networks. Future proxy designs should address and rectify this limitation.

More importantly, as highlighted earlier, the majority of existing zero-shot proxies (with the exceptions of NTK_Cond and NN-Mass) concentrate solely on one of three dimensions: {expressive capacity, generalization capacity, trainability}. This is a fundamental shortcoming, as a good neural network seamlessly integrates all three facets. We provide empirical evidence of this concern in Section 4.

3 Benchmarks and Profiling Models

NAS benchmarks have been proposed to provide a standard test kit for fair evaluation and comparisons of various NAS approaches [98], [99], [100], [101]. A NAS benchmark defines a set of candidate architectures and their test accuracy or hardware costs. We classify the existing NAS benchmarks as standard NAS (*i.e.,* without hardware costs) and hardware-aware NAS benchmarks. Next, we introduce these two types of NAS benchmarks.

3.1 NAS Benchmarks

We evaluate the zero-shot proxies on the following standard NAS benchmarks: **NASBench-101** provides 423k neural architectures and their test accuracy on the CIFAR10 dataset, where each architecture is built by stacking a cell for multiple times [102]. **NATS-Bench** contains two sub-search spaces: *(i) NATS-Bench-TSS*, also known as NASBench-201; each network in NASBench-201 is also built by repeating a cell multiple times on three datasets, namely, CIFAR10, CIFAR100, and ImageNet16-120 [103] (see Figure 5 for more details); *(ii) NATS-Bench-SSS* contains 32768 architectures with different width values for each layer [104][2]. **TransNAS-Bench-101** is

1. A cell represents a group of layers with the same width values or commonly used blocks in CNN, *e.g.,* Basic/Bottleneck blocks in ResNet, and Inverted bottleneck blocks in MobileNet-v2.

2. In the rest of the paper, we use the NATS-Bench to represents NATS-Bench-SSS for short.

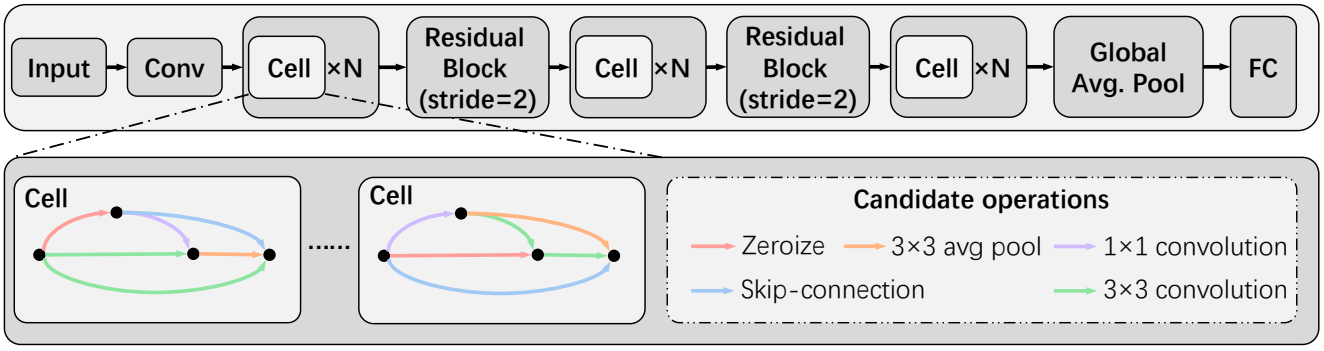


Fig. 5: Search space of NASBench-201. Each architecture in the search space is built by stacking a cell multiple times; each cell can have six operations (edges in the figure) and each operation has 5 potential different options (drawn with different colors). NASBench-101 has a very similar search space with more candidate operations. (Adapted from [11])

a benchmark dataset containing network performance on seven diverse vision tasks, including image classification, image reconstruction, and pixel-level prediction [105] with two different sub-search spaces: (*i*) A cell-level search space consisting of 4,096 unique networks with different cells; (*ii*) A macro-level search space containing 3256 unique networks with different depth values.

Hardware-aware NAS benchmarks. Recent hardware-aware NAS approaches aim to jointly optimize the test performance and hardware efficiency of neural architectures. Hence, hardware-aware NAS benchmarks have been proposed by incorporating the hardware costs of networks into the search process. HW-NAS-Bench covers the search space from both the NASBench-201 and FBNet [106]. It provides all the architectures in these two search spaces measured/estimated hardware cost (*i.e.,* latency and energy consumption) on multiple types of devices. Similarly, Eagle, also known as BRP-NAS, provides a benchmark that contains latency and energy for NAS-Bench-201 networks running on up to 13 devices spanning a wide spectrum from the cloud server to the edge devices; this ameliorates the need for researchers to have access to these devices [107]. Moreover, Eagle also proposes an efficient performance estimator for measuring and predicting the performance of neural networks (cf. Section 3.2).

3.2 Hardware Performance Models

To incorporate the hardware-awareness into NAS, we also need to construct models to efficiently and accurately estimate the hardware performance (*e.g.,* latency) of given networks. In this section, we consider latency to characterize the hardware performance and use NASBench-201 as an example to compare several representative approaches for hardware performance models.

BRP-NAS is a pioneering approach that uses deep learning to build hardware performance models [107]. Specifically, BRP-NAS first converts a neural network into a directed acyclic graph by modeling each layer as an edge in a graph and modeling the input/output as nodes in the graph. Next, by using different values to present different types of layers, BRP-NAS uses a Graph Convolution Network (GCN) to build the hardware performance models. Then the model is trained with multiple networks and their real hardware performance data on the target hardware. In particular, for the networks with fixed depth, BRP-NAS can also use MLP to build the performance model. Though BRP-NAS can achieve good prediction results with enough training samples, there is a limitation for BRP-NAS: the performance model is trained for a specific hardware platform; if new hardware comes, one needs to repeat the entire process.

TABLE 3: Comparison of representative hardware performance models. Granularity refers to the level of input features for the hardware performance models, and transferability denotes the efficiency with which the model for one hardware platform can be transferred to another. The latency is measured on Snapdragon-888's GPU with NASBench-201 on CIFAR100 dataset.

Approach	Method	Granularity	Transferability	RMSE(ms)
BRP-NAS [107]	GCN or MLP	Layer	Low	4.6
HELP [108]	GCN or MLP	Layer	High	0.12
NN-Meter [109]	GCN	Kernel	Low	1.2

To address the above problem, HELP builds the hardware performance models by taking the hardware information as extra input features (*e.g.,* type of the hardware, number of computing elements, and the size of on-chip memory) [108]. Next, HELP is trained with the latency data collected from multiple platforms, such as desktop CPU/GPU and mobile CPU/GPU. This way, if new hardware comes in, HELP only needs a few samples to conduct the fine-tuning process (typically around 10). Hence, HELP is very efficient in terms of the transferability to new hardware. Nevertheless, both BRP-NAS and HELP are built on the layer-level analysis, which is relatively coarse for an accurate prediction.

To further improve the accuracy of performance models, NN-Meter is proposed by analyzing the neural network at a finer granularity during run-time. Specifically, NN-Meter computes the kernels of each neural network, which are originally generated during the compilation process [109]. To remove the necessity of the compilation process, NN-Meter utilizes the algorithm to automatically predict the generated kernels. Hence, as shown in Table 3, NN-Meter has a much higher prediction quality than both HELP and BRP-NAS.

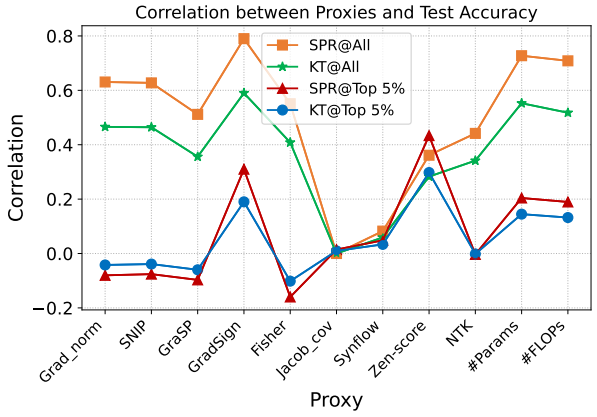


Fig. 6: The correlation between various proxies and the test accuracy on NASBench-201 search space for CIFAR-100 dataset (averaged over 5 seeds). All: all the networks in the benchmark; Top 5%: the architectures with test accuracy ranking top 5% in the entire search space. KT and SPR are short for Kendall's τ and Spearman's ρ, respectively (same for other figures).

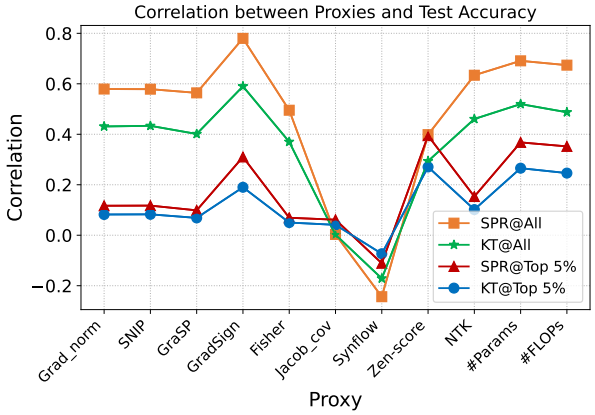


Fig. 7: The correlation between various proxies and the test accuracy on NASBench-201 search space for ImageNet16-120 dataset (averaged over 5 seeds).

4 Experimental Results

In this section, we compare the existing proxies on multiple NAS benchmarks under various scenarios. Besides the proxies mentioned above, we also evaluate two naive proxies, *i.e.*, #Params and #FLOPs.

Evaluation Metrics. We use two commonly used criteria to evaluate the correlations between different zero-shot proxies and their test accuracies across different benchmarks:

- Spearman's ρ. Spearman's ρ quantifies the monotonic relationships between two variables within the range of [-1, 1], where $\rho = 1$ indicates a perfect positive correlation between these two variables, while $\rho = -1$ indicates a perfect negative correlation. We use "SPR" for short to represent Spearman's ρ in the tables and figures of this paper.
- Kendall's τ. Similar to Spearman's ρ, Kendall's τ value is also within [-1, 1]. Typically, Kendall's τ is more robust to error and discrepancies than Spearman's ρ. We use "KT" for short to represent Kendall's τ in the tables and figures of this paper.

In NAS, the architectures with good performance are more important than those with poor performance. Hence, we also calculate Spearman's ρ and Kendall's τ for the architectures with test accuracy ranking top 5% in the entire search space, which are denoted as "SPR@Top 5%" and "KT@Top 5%", respectively. Similarly, if we calculate Spearman's ρ and Kendall's τ for all architectures in the search space, they are denoted as "SPR@All" and "KT@All", respectively.

4.1 NAS without hardware-awareness

To compare the performance of these proposed accuracy proxies, we calculate the correlation of these proxy values and the real test accuracy. We next discuss the results on two NAS benchmarks: NASBench-201 and NATS-Bench.

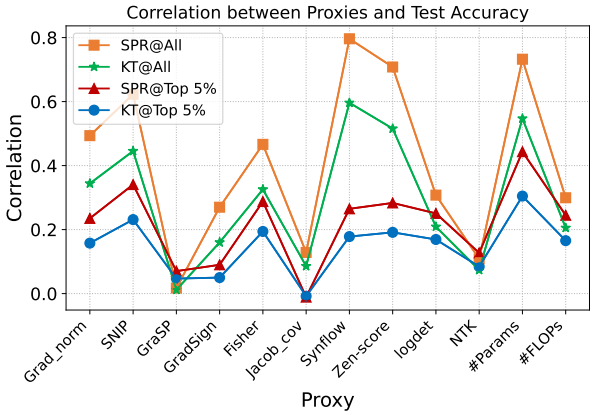


Fig. 8: The correlation between various proxies and the test accuracy on NATS-Bench search space for CIFAR100 dataset (averaged over 5 seeds).

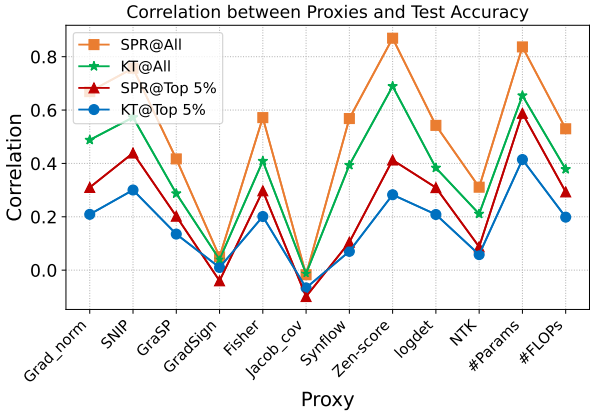


Fig. 9: The correlation between various proxies and the test accuracy on NATS-Bench search space for ImageNet16-120 dataset (averaged over 5 seeds).

TABLE 4: The test accuracy (%) of optimal architectures obtained by various zero-shot proxies (averaged over 5 runs) on NASBench-201 (NB201) and NATS-Bench (NB201) for CIFAR100 (C100) and ImageNet16-120 (Img16) datasets. The best results are shown with bold fonts.

	Proxies	Ground Truth	Grad_norm	SNIP	GraSP	GradSign	Fisher	Jacob_cov	Synflow	Zen-score	#Params	#FLOPs
NB201	C100	73.51	60.02	60.02	60.02	60.02	60.02	68.89	62.22	68.10	**71.11**	**71.11**
	Img16	47.31	29.27	29.27	5.46	5.46	29.27	25.07	26.08	40.77	**41.44**	**41.44**
NATS	C100	70.92	48.44	68.36	57.40	57.40	53.14	55.04	66.84	69.92	**70.28**	**70.28**
	Img16	46.73	40.97	45.63	33.97	33.97	35.80	35.03	35.37	**46.27**	44.73	44.73

4.1.1 Unconstrained search space

We first investigate the performance of zero-shot proxies for the unconstrained search spaces, *i.e.,* considering all networks in the benchmarks.

NASBench-201: We calculate the correlation coefficients between multiple proxies and the test accuracy on CIFAR-100 and ImageNet16-120 datasets. As shown in Figure 6 and 7, the #Params generally works best for these two datasets. Except for the #Params, several gradient-based proxies, such as Grad_norm, SNIP, GraSP, and Fisher, also work well.

As shown in Table 4, we compare the neural architectures with the highest test accuracy found via various proxies. The neural architectures obtained via #Params and #FLOPs have the highest test accuracy on NASBench-201, which is natural and expected results given the correlation scores above.

NATS-Bench: Similar to NASBench-201, we calculate the correlation coefficients between these proxies and the test accuracy on CIFAR-100 and ImageNet16-120 datasets for NATS-Bench. As shown in Figure 8 and 9, the #Params and Zen-score generally work best for these two datasets.

TransNAS-Bench-101: So far, we primarily compare these zero-shot proxies on the classification tasks. To verify the effectiveness of these proxies for more diverse applications, we make comparisons for non-classification tasks selected from the TransNAS-Bench-101. We pick the largest search space TransNAS-Bench-101-Micro which contains 4096 total architectures with different cell structures. We compare these proxies under the following three non-classification tasks:

- Semantic segmentation. Semantic segmentation involves classifying each pixel in an image into a predefined category or class. Unlike object detection, which identifies the bounding boxes around objects, or image classification, which assigns a single label to the entire image, semantic segmentation provides a detailed, pixel-level classification.
- Surface Normal. Similar to semantic segmentation, surface normal is a pixel-level prediction task that predicts surface normal statistics.
- Autoencoding. Autoencoding is an end-to-end image reconstruction task that encodes an input image into a low-dimension representation vector and then reconstructs this vector into the input image.

As shown in Figure 10, Jacob_cov typically achieves the highest correlation for these two tasks and consistently outperforms #Params. Besides Jacob_cov, the Zen-score also works well and it consistently surpasses #Params.

We also compare the neural architectures with the highest test accuracy found via various proxies. As shown in Table 5, the neural architectures obtained via #Params and #FLOPs consistently have the highest or second-highest test performance on TransNAS-Bench-101.

Overall, it appears that none of these proposed accuracy proxies consistently have a higher correlation with the test accuracy compared to #Params and #FLOPs for these two NAS benchmarks.

4.1.2 Constrained search space

We note that the architectures with high accuracy are much more important than those networks with low test accuracy. Hence, we calculate the correlation coefficient for the architectures with test accuracy ranking top 5% in the entire search space. Figure 6 and 7 show that, compared to ranking without constraints (*i.e.,* considering all architectures), the correlation score has a significant drop except for the Zen-score on NASBench-201. Similarly, on NATS-Bench, Figure 8 and 9 show that most of the proxies have a significant correlation score drop when constrained to the top 5% networks in the search space, including #Params and #FLOPs. By switching to non-classification tasks, we observe a similar trend in Figure 10, i.e., there's a significant correlation score drop under these constrained scenarios.

This drop in correlation score for the top 5% of networks means the zero-shot NAS is more likely to miss the optimal or near-optimal networks. Table 4 shows that there is a big accuracy gap between the ground truth and the networks obtained by each proxy. results become even worse with a search that has more relaxed hardware constraints (see Sec 4.4).

As shown in previous literature, #Params and #FLOPs outperform other proxies in multiple benchmarks [63]. Hence, we dig deep into the effectiveness of #Params and #FLOPs by gradually making the search space more constrained. As shown in Figure 13 and Figure 14, if we compute the correlation for networks with higher accuracy, both #Params and #FLOPs have a significant drop in correlation score.

Given the above results, we conclude that all of the existing proxies (including #Params and #FLOPs) do *not* correlate well for the network with high accuracy. This is a fundamental drawback because what matters most for NAS are precisely these networks with high accuracy. Hence, there is great potential for designing better proxies that could yield high correlation scores for these top networks.

4.1.3 Specific Network Families

We remark that many popular neural architectures are not included in most NAS benchmarks. Hence, in this section, we consider several commonly used network families as the search space since they are widely used in various applications. As shown in Figure 11, if we search within

TABLE 5: The test performance of optimal architectures obtained by various zero-shot proxies (averaged over 5 runs) on TransNAS-Bench-101 benchmarks. The best results are shown with bold fonts. Here, the evaluation metric for semantic segmentation is mIoU, while the rest two use SSIM [110].

Task	GroundTruth	Gradnorm	SNIP	GraSP	GradSign	Fisher	Jacob_cov	Synflow	Zen-score	#Params	#FLOPs
Semantic Segmentation	94.61	91.66	94.43	94.53	90.19	91.89	94.34	94.46	**94.50**	**94.50**	**94.50**
Surface Normal	0.59	0.53	0.53	0.38	**0.57**	**0.57**	0.55	0.53	0.55	0.55	0.55
Autoencoding	0.58	0.36	0.33	0.33	0.35	**0.49**	0.42	0.46	0.46	0.46	0.46

TABLE 6: Comparison of zero-shot proxies based NAS vs. one-shot NAS on ProxylessNAS search space. The results are averaged over three runs.

Method	One-shot NAS	Grad_norm	Synflow	GradSign	Jacob_cov	NTK_Cond	Zen-score	Params	FLOPs
Top-1 on ImageNet-1K	74.39	71.46	70.02	73.17	70.31	73.63	71.78	72.87	73.08
mAP on COCO	0.28	0.22	0.21	0.23	0.24	0.27	0.25	0.26	0.28
Search cost (GPU Hours)	200	8.9	8.8	9.7	9.2	37	1.6	0.03	1.5

networks from ResNet and Wide-ResNet families, then SNIP, Zen-score, #Params, #FLOPs, and NN-Mass have a significantly high correlation with the test accuracy (*i.e.*, Spearman's $\rho > 0.9$).

As shown in Figure 12, Grad_norm, SNIP, Fisher, Synflow, Zen-score, and NN-Mass work best for the MobileNet-v2 network family, which is slightly better than the two naive proxies #Params and #FLOPs. These results show that there is great potential in designing good proxies for a constrained yet widely used search space.

4.2 Large-scale Dataset

To further compare these proxies in more complicated scenarios, we illustrate the performance for ImageNet-1K classification, COCO object detection, and ADE20K semantic segmentation tasks.

ImageNet-1K classification. We first compute the zero-shot proxies for the CNN architectures in the model space of TIMM [111]. Notably, we only consider networks that are trained standalone on ImageNet-1K without pre-training or distillation. In total, we evaluate 200 CNNs and report the correlation between Top-1 accuracy and multiple proxies in Figure 15. As shown, #Params and #FLOPs still have a higher correlation than these zero-shot proxies. This is consistent with our observations on NAS benchmarks.

We also compare the performance of these proxy-based NAS with one-shot NAS within the same search space. We conduct the comparison on the MobileNet-V2 based search space under the same #FLOPs budget of 600M. Specifically, for proxy-based NAS, we use the evolutionary algorithm to search for the architecture with the highest proxy values; we conduct the search for at most 10K steps. For the one-shot NAS, we use the same algorithm from [36]. We train the obtained architecture for 150 epochs under the standard data augmentation configurations. We use the SGD optimizer with an initial learning rate of 0.1 and a cosine annealing learning rate schedule.

As shown in Table 6, compared to one-shot NAS, zero-shot proxy-based NAS has a slight accuracy degradation (less than 1%), but requires orders of magnitude less search costs. Moreover, when comparing these zero-shot proxies, NTK_Cond based search performs closest to one-shot NAS, but at a higher search cost than other proxies. These results highlight an intrinsic trade-off between search cost and the accuracy of the obtained architectures.

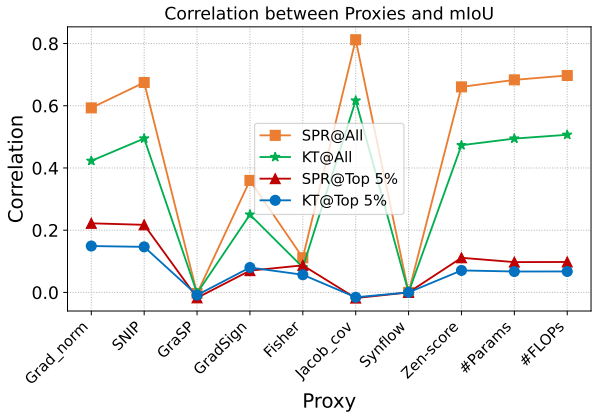
COCO object detection. Following the standard practice in NAS, we employ the architectures obtained on ImageNet-1K (shown in Table 6) as the backbone for detection models. By using the detection head from NanoDet [112], we then train these networks for 50 epochs on COCO following the same training setup as NanoDet. As shown in Table 6, the results follow a trend similar to that of ImageNet-1K. More precisely, #FLOPs and NTK_Cond based zero-shot NAS yield performance that is the same or very close to the one-shot NAS.

ADE20K semantic segmentation. We compute these zero-shot proxies for the CNN architecture in the model space of PyTorch Segmentation [111]. We vary both the backbone and segmentation heads to obtain multiple segmentation networks; we then train these models from scratch and get their test performance. In total, we evaluate 200 CNNs and report the correlation between pixel accuracy (or mIoU) and various proxies in Figure 16. As shown, #Params and #FLOPs have a higher correlation than the other zero-shot proxies.

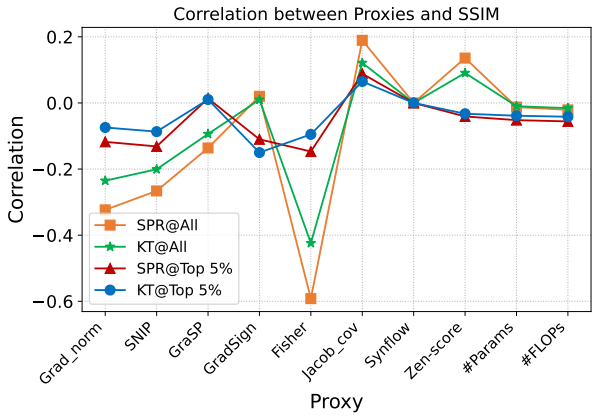
To conclude, these comprehensive evaluations on these large-scale datasets reaffirm the dominance of #Params and #FLOPs over other proxies in multiple scenarios. Therefore, future works should make comprehensive comparisons under various tasks and datasets to show a consistent advantage over #Params and #FLOPs. Besides, while zero-shot proxy-based NAS exhibits certain efficiencies, there remains a trade-off between search cost and test performance accuracy.

4.3 Vision Transformers

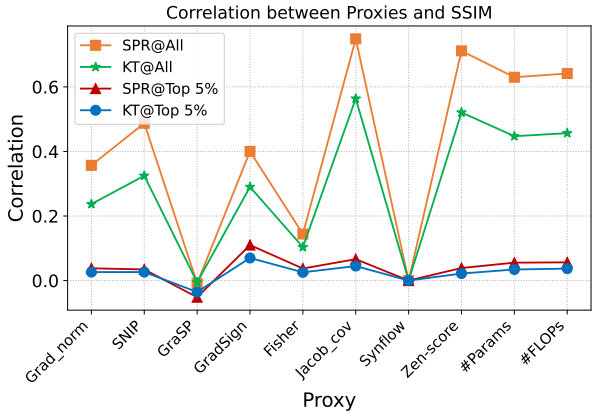
Until now, our evaluations have primarily focused on CNNs; however, with the recent surge in their performance and popularity, vision transformers (ViTs) are becoming increasingly important in the realm of computer vision [5]. Therefore, in this section, we evaluate these proxies using the ViT model space for ImageNet-1K.



(a) Semantic Segmentation



(b) Autoencoding



(c) Surface Normal

Fig. 10: The correlation between various proxies and test performance on TransNAS-Bench-101 for Semantic Segmentation, Autoencoding, and Surface Normal tasks (averaged over 5 seeds).

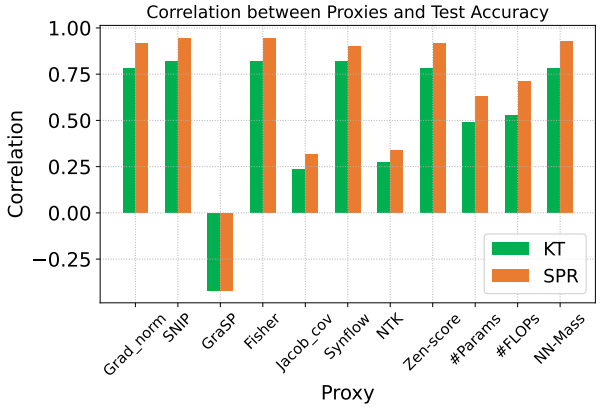


Fig. 11: The correlation between various proxies and the test accuracy on a set of ResNets and Wide-ResNets for ImageNet-1K classification (averaged over 5 seeds).

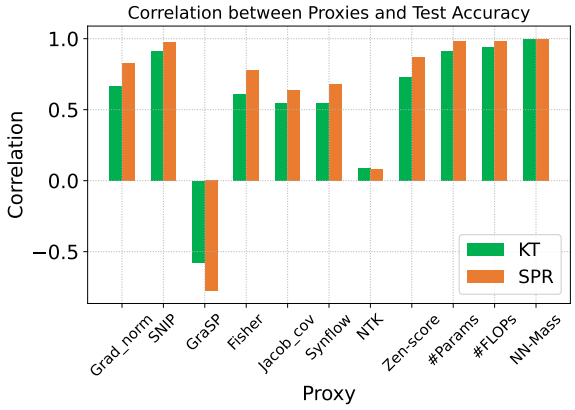


Fig. 12: The correlation between various proxies and the test accuracy on a set of MobileNet-v2-based networks for ImageNet-1K classification (averaged over 5 seeds).

Specifically, we compare these zero-shot proxies for the ViTs in the model space of TIMM [111]. Notably, we only include networks that are trained standalone on ImageNet-1K without pre-training or distillation. In total, we evaluate 100 ViTs and report the correlation between Top-1 accuracy and various proxies in Figure 17. The results show that #Params and #FLOPs has higher correlation score with the test accuracy than the zero-shot proxies. This is consistent with our observations on CNNs. In conclusion, whether analyzing CNNs or ViTs, the superior correlation of #Params and #FLOPs over zero-shot proxies is consistent.

In practical applications, test performance is not the only design consideration. Indeed, the models obtained by NAS should meet some hardware constraints, especially for deployment on edge devices. Hence, we next explore the performance of these proxies for the hardware-aware search scenarios.

4.4 Hardware-aware NAS

In this part, we conduct the hardware-aware NAS using the zero-shot proxies introduced above. Specifically, we use these

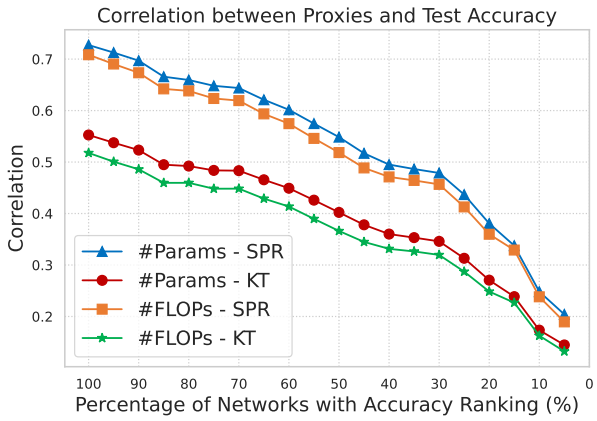


Fig. 13: The correlation between #Params & #FLOPs and the test accuracy under various ratios of networks on NASBench-201 for CIFAR100 dataset (averaged over 5 seeds). 20% means computing the correlation scores only for the networks whose test accuracy ranks top 20% in the benchmark; 100% means considering all the networks in the benchmark (same for Figure 14). From left to right, the search space is more and more constrained to neural architectures with high accuracy.

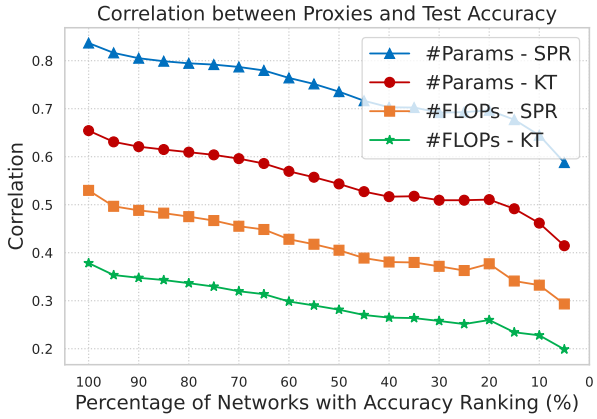


Fig. 14: The correlation between #Params & #FLOPs and the test accuracy under various ratios of networks on NATS-Bench for ImageNet16-120 dataset (averaged over 5 seeds).

zero-shot proxies instead of the real test accuracy to search for the Pareto-optimal networks under various constraints. We next introduce the results on NASBench-201 (with HW-NAS-Bench) and NATS-Bench.

4.4.1 *NASBench-201 / HW-NAS-Bench*

We use EdgeGPU (NVIDIA Jetson TX2) as the target hardware and use the energy consumption data from HW-NAS-Bench; then we set various energy consumption values as the hardware constraints. Next, we use different accuracy proxies to traverse all candidate architectures in the search space and obtain the Pareto-optimal networks under various energy constraints.

To illustrate the quality of these networks, we plot these networks and the ground truth results obtained via actual accuracy in Figure 18. As shown, when the energy constraint

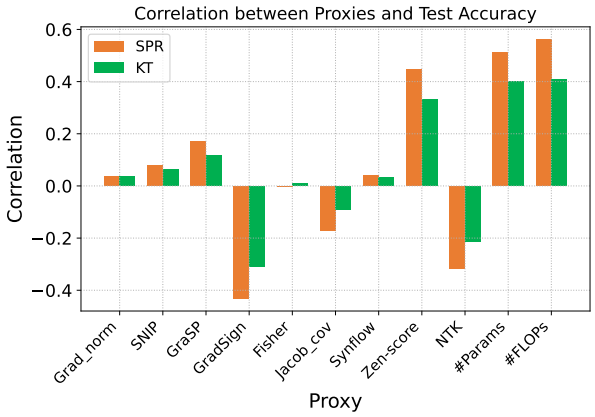
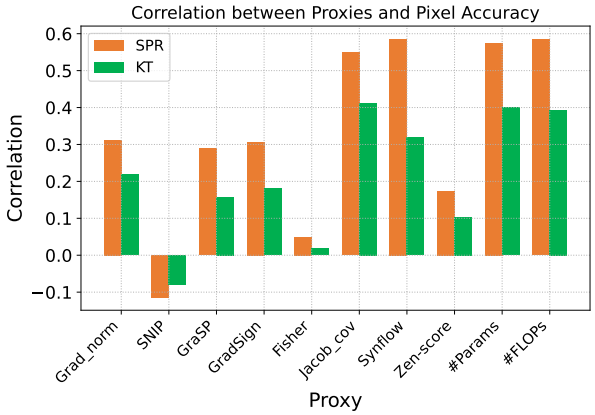
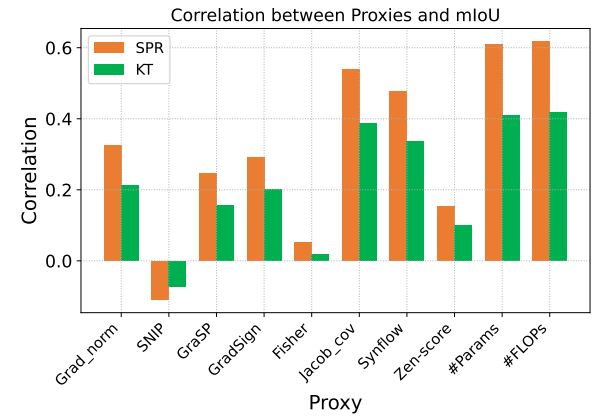


Fig. 15: The correlation between various proxies and the test accuracy on the CNNs model space for ImageNet-1K classification.



(a) Pixel Accuracy



(b) mIoU

Fig. 16: The correlation between various proxies and pixel accuracy (or mIoU) on ADE20K semantic segmentation.

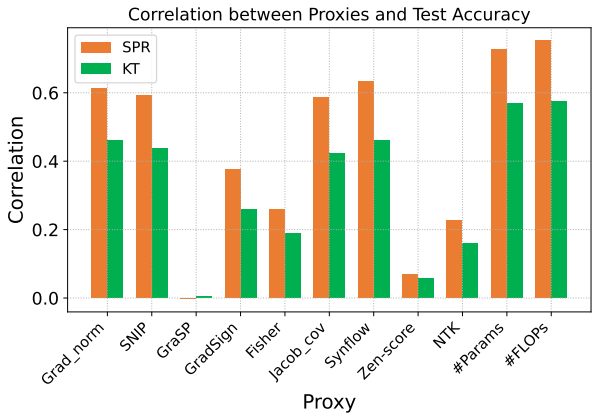


Fig. 17: The correlation between various proxies and the test accuracy on the ViT model space for ImageNet-1K classification.

is tight (*e.g.,* less than 10mJ), most of the proxies could find networks very close to the real Pareto-optimal, except the Jacob_cov. However, when the energy constraint is more relaxed (*e.g.,* more than 20mJ), only #Params, #FLOPs, and Jacob_cov can find several networks close to the ground truth.

4.4.2 NATS-Bench

We measure the latency data on NVIDIA GTX-1080 for NATS-Bench. We then use different accuracy proxies to traverse all candidate architectures to obtain the Pareto-optimal networks under various latency constraints. As shown in Figure 19, we plot these networks and the ground truth results. When we set the latency constraint to around 50ms, only #Params, SNIP, and Zen-score can still find the networks that nearly match the real Pareto-optimal networks.

The results on these two benchmarks further verify that current proxies don't correlate well for networks with high accuracy because the real Pareto-optimal networks have higher accuracy when the hardware constraints are more relaxed. This observation suggests a great potential to design better proxies in this scenario.

4.5 Discussion and future work

4.5.1 NAS Benchmarks

Diversity of search space: We remark that the search space of most existing NAS benchmarks only contains cell-based neural architectures. To further improve the generality of NAS benchmarks, the community may need to incorporate new architectures from more diverse search spaces. For instance, the NATS-Bench has added architectures with different cells for different stages of the search space. Moreover, the cells in these existing benchmarks are similar to the DARTS cell structure. However, in practice, the inverted bottleneck blocks from MobileNet-v2 are more widely used for higher hardware efficiency. Therefore, the next direction of NAS benchmarks may need to cover a more practical and widely used search space, such as FBNet-v3.

Awareness of hardware efficiency: So far, only HW-NAS-Bench provides multiple hardware constraints on several types of hardware platforms, but it does not have the accuracy data for most of the networks in the benchmark. Thus, we recommend future NAS benchmarks to incorporate both accuracy and hardware metrics for typical hardware platforms.

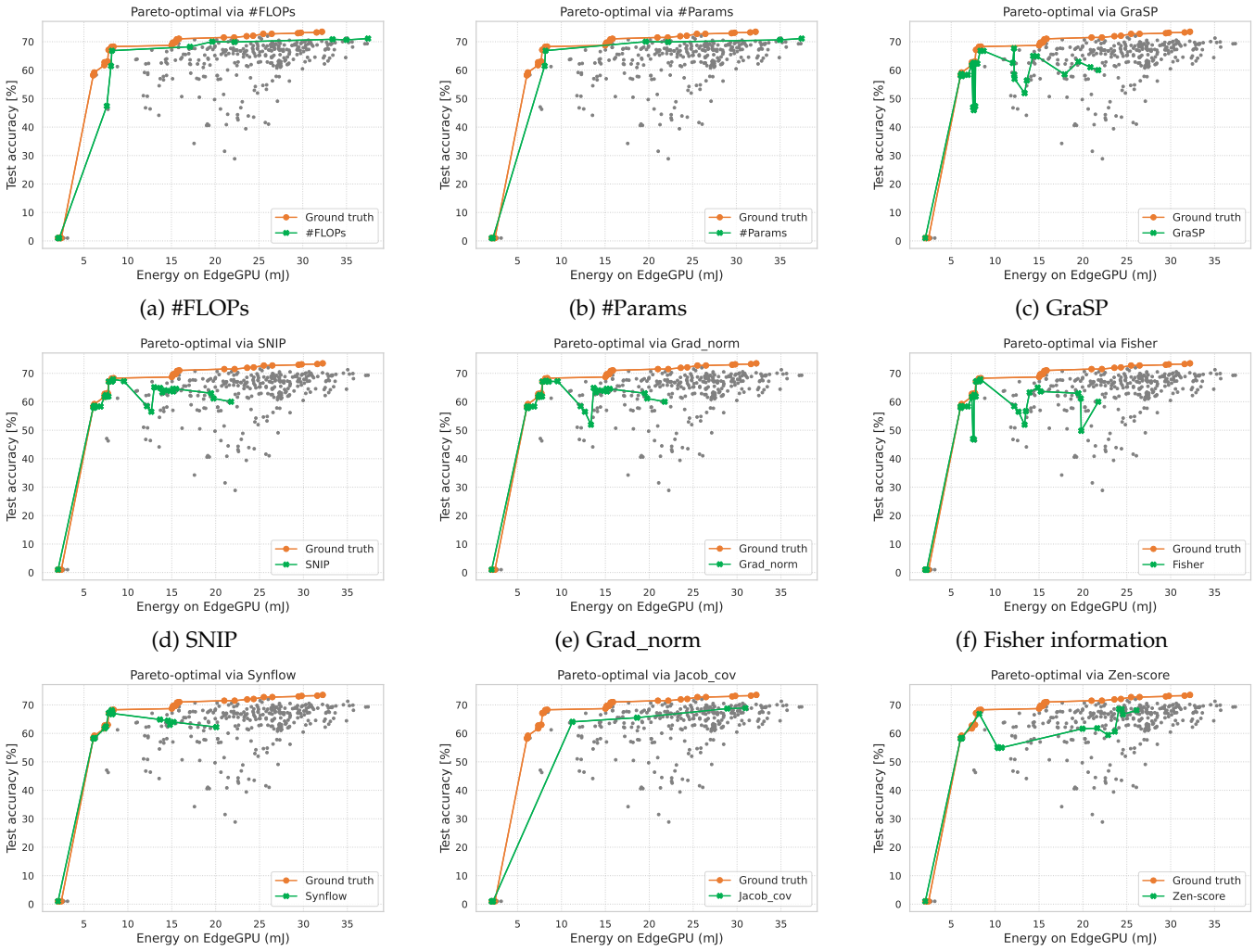
4.5.2 Zero-shot proxies

Why #Params works: As shown in Section 4.1.1, #Params achieves a higher correlation than other proxies with multiple datasets and multiple benchmarks for unconstrained search space. One may wonder why such a trivial proxy works so well. In general, a good neural architecture should satisfy the following properties: good convergence/trainability and high expressive capacity. We provide the following observations:

- **Expressive Capacity** It is well known that a network with infinite width or depth, can express any type of complex functions with an arbitrarily small errors [113], [114], [115]. Moreover, previous works show that, with the depth or width values increasing, the error w.r.t. ground truth functions will gradually decrease. In other words, more parameters capture the higher expressive capacity of a given neural network [68].
- **Generalization Capacity** Previous work reveals that a network with more parameters tends to have higher test accuracy under an appropriate training setup [116].
- **Trainability** On the one hand, given similar depth, the wider networks have better trainability and higher convergence rates, and clearly more parameters [53]. On the other hand, most of the networks evaluated on popular benchmarks share a similar depth value. Hence, within these benchmarks, more parameters will also indicate a better trainability.

Hence, #Params captures both the expressivity and trainability of the networks in these benchmarks. In contrast, most of the proposed proxies usually emphasize either the expressivity or the trainability of networks (but not both). That may be why #Params outperforms these proposed proxies. Hence, future work should aim to design a proxy that could indicate both the convergence/trainability and expressive and generalization capacity of a given network. For instance, recently proposed proxy ZiCo indicates both trainability and generalization capacity of neural networks thus consistently outperforming #Params in multiple NAS benchmarks [117].

When #Params fails: (*i*) As shown in this section, when accounting for the architectures with test accuracy ranking top 5%, several proxies outperform both #Params and #FLOPs for some benchmarks. Furthermore, these top-performing network architectures are most important since NAS focuses on obtaining the networks with high accuracy. (*ii*) Many proxies work well in the constrained search space, such as the MobileNet and ResNet families. These networks are widely used in many applications (*e.g.,* MobileNet-v2 for EdgeAI). Clearly, the above two failing cases are very important to push zero-shot NAS to more practical scenarios. Hence, there is a great potential to explore better zero-shot proxies in the above cases.



(a) #FLOPs (b) #Params (c) GraSP

(d) SNIP (e) Grad_norm (f) Fisher information

(g) Synflow (h) Jacob_cov (i) Zen-score

Fig. 18: Pareto-optimal networks obtained via various proxies for CIFAR100 dataset on NASBench-201, and for various energy consumption constraints on an EdgeGPU (NVIDIA Jetson TX2). The gray points in these figures are candidate networks in the search space.

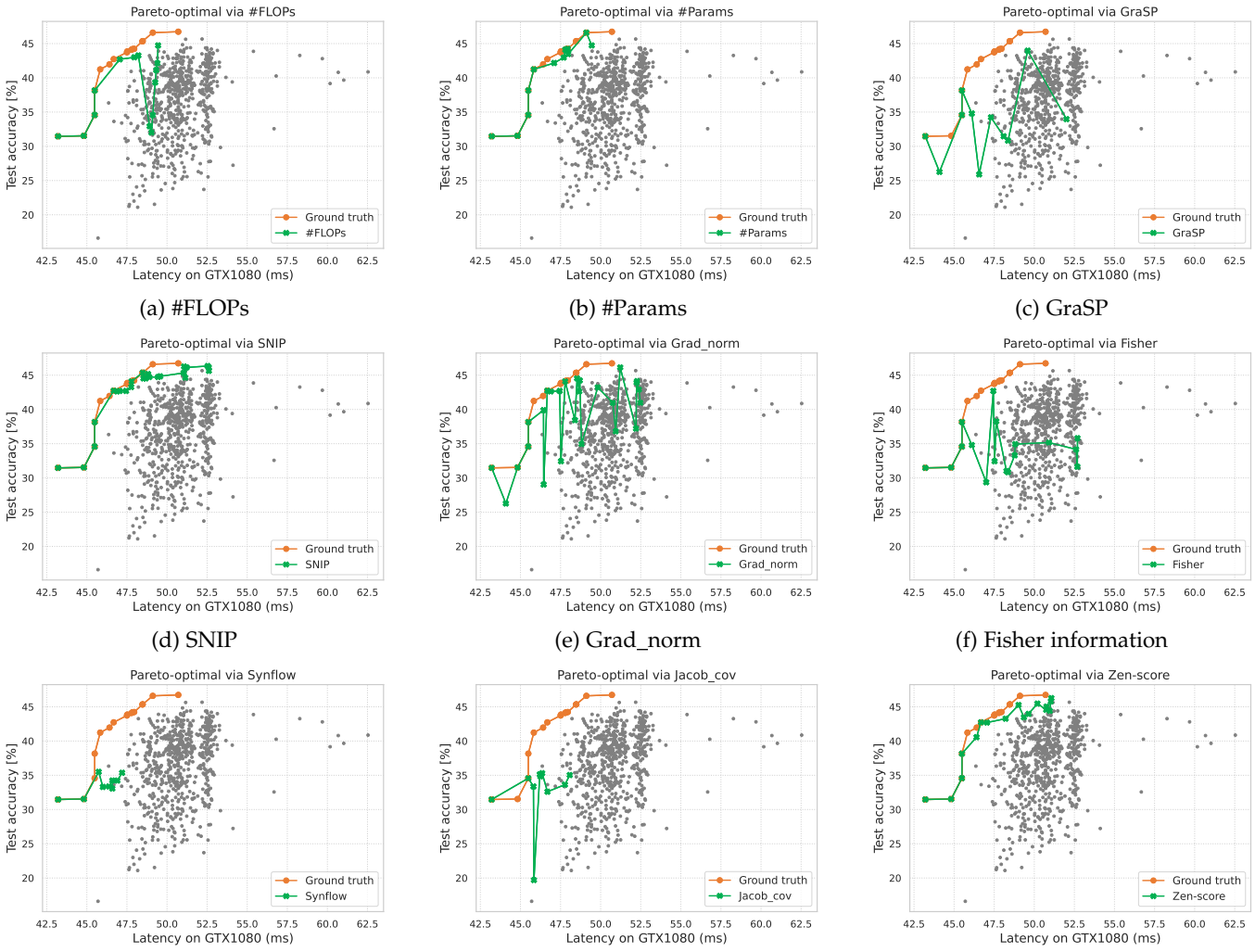
Search method: Though #Params outperforms most proxies in several scenarios in terms of correlation coefficients, there are alternative search methods to use these zero-shot proxies. For example, as demonstrated in [54], to better leverage these proxies, one potential search method can merge all candidate networks into a supernet and then apply these proxies to prune the network at the initialization stage until hardware constraints are met. This way, the time efficiency of zero-shot NAS approaches can be further improved since the search space is gradually compressed with pruning going on.

Theoretical support: We remark that most gradient-based proxies are first proposed to estimate the importance of each parameter or neuron/channel of a given network, thus originally applied to the model pruning problem space instead of ranking networks. Hence, the effectiveness of these gradient-based proxies for zero-shot NAS needs a more profound understanding from a theoretical perspective. Moreover, though most gradient-free proxies are usually presented with some theoretical analysis for NAS, as shown in Section 4.1 and Section 4.4, they generally have a lower correlation with the gradient-based ones. The theoretical understanding of why these zero-shot proxies can or cannot estimate the test accuracy of different networks is still an open question.

Customized proxy for different types of networks: As mentioned in Section 4.1.3, several zero-shot proxies do not work well for a general search space, but do show a great correlation with the test accuracy and beat the #Params on constrained search spaces. In fact, Section 4.1 and Section 4.4 show that designing a zero-shot proxy that generally works well is extremely difficult. One potential direction for the design of zero-shot proxies may lie in partitioning the entire search space into several sub-spaces and then proposing customized proxies specifically designed for different sub-spaces.

5 Conclusion

In this paper, we have presented a comprehensive review of existing zero-shot NAS approaches. To this end, we have first introduced accuracy proxies for zero-shot NAS by providing theoretical inspirations behind these proxies,



(a) #FLOPs (b) #Params (c) GraSP

(d) SNIP (e) Grad_norm (f) Fisher information

(g) Synflow (h) Jacob_cov (i) Zen-score

Fig. 19: Pareto-optimal networks obtained via various proxies for ImageNet16-120 dataset on NATS-Bench, and for various latency constraints on NVIDIA GTX1080. The gray points in these figures are candidate networks in the search space.

and several commonly used NAS benchmarks. We then have introduced several popular approaches for hardware performance predictions. We have also compared the existing proxies against two naive proxies, namely, #Params and #FLOPs. By calculating the correlation between these proxies and the real test accuracy, we have shown that the proposed proxies to date are not necessarily better than #Params and #FLOPs for these tasks for unconstrained search spaces (*i.e.*, considering all architectures in benchmarks). However, for constrained search spaces (*i.e.*, when considering only networks with high accuracy), we have revealed that the existing proxies, including #Params and #FLOPs, has much worse correlation scores with the real accuracy than unconstrained scenarios. Based on these analyses, we have explained why #Params work and when #Params fail. Finally, we have pointed out several potential research directions to design better benchmarks for better zero-shot NAS and multiple ideas that may enable the design of better zero-shot NAS approaches.

ACKNOWLEDGMENTS

Radu Marculescu and Guihong Li are supported in part by the NSF grant CNS 2007284, and in part by the iMAGiNE Consortium [Link]. Z. Wang is in part supported by NSF Scale-MoDL (#2133861).

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in Neural Information Processing Systems, 2012.

[2] S. Liu and W. Deng, "Very deep convolutional neural network based image classification using small training sample size," in 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR), 2015.

[3] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[5] A. Dosovitskiy et al., "An image is worth 16x16 words: Transformers for image recognition at scale," in International Conference on Learning Representations, 2021.

[6] T. B. Brown et al., "Language Models are Few-Shot Learners," arXiv preprint arXiv:2005.14165, 2020.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," Advances in neural information processing systems, vol. 30, 2017.

[8] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing Neural Network Architectures using Reinforcement Learning," arXiv preprint arXiv:1611.02167, 2016.

[9] B. Zoph and Q. V. Le, "Neural Architecture Search with Reinforcement Learning," arXiv preprint arXiv:1611.01578, 2016.

[10] C. Liu et al., "Progressive Neural Architecture Search," in Proceedings of the European Conference on Computer Vision (ECCV), 2018.

[11] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," arXiv preprint arXiv:1806.09055, 2018.

[12] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," The Journal of Machine Learning Research, 2019.

[13] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient Neural Architecture Search via Parameters Sharing," in International Conference on Machine Learning. PMLR, 2018, pp. 4095–4104.

[14] E. Real et al., "Large-scale Evolution of Image Classifiers," in International Conference on Machine Learning. PMLR, 2017.

[15] X. Gong, S. Chang, Y. Jiang, and Z. Wang, "Autogan: Neural architecture search for generative adversarial networks," in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 3224–3234.

[16] S. Xie, H. Zheng, C. Liu, and L. Lin, "SNAS: stochastic neural architecture search," in International Conference on Learning Representations, 2019.

[17] B. Wu et al., "Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019.

[18] A. Wan, X. Dai, P. Zhang, Z. He, Y. Tian, S. Xie, B. Wu, M. Yu, T. Xu, K. Chen et al., "Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 12 965–12 974.

[19] L. Li and A. Talwalkar, "Random search and reproducibility for neural architecture search," in Uncertainty in artificial intelligence. PMLR, 2020, pp. 367–377.

[20] K. Kandasamy, W. Neiswanger, J. Schneider, B. Poczos, and E. P. Xing, "Neural architecture search with bayesian optimisation and optimal transport," Advances in neural information processing systems, vol. 31, 2018.

[21] K. Yu, C. Sciuto, M. Jaggi, C. Musat, and M. Salzmann, "Evaluating the search phase of neural architecture search," arXiv preprint arXiv:1902.08142, 2019.

[22] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search," arXiv preprint arXiv:1711.00436, 2017.

[23] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang, "Efficient architecture search by network transformation," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, no. 1, 2018.

[24] R. Luo, F. Tian, T. Qin, E. Chen, and T.-Y. Liu, "Neural architecture optimization," Advances in neural information processing systems, vol. 31, 2018.

[25] C. Zhang, M. Ren, and R. Urtasun, "Graph hypernetworks for neural architecture search," arXiv preprint arXiv:1810.05749, 2018.

[26] H. Zhou, M. Yang, J. Wang, and W. Pan, "Bayesnas: A bayesian approach for neural architecture search," in International conference on machine learning. PMLR, 2019, pp. 7603–7613.

[27] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan et al., "Searching for mobilenetv3," in Proceedings of the IEEE/CVF international conference on computer vision, 2019, pp. 1314–1324.

[28] J. Yu, P. Jin, H. Liu, G. Bender, P.-J. Kindermans, M. Tan, T. Huang, X. Song, R. Pang, and Q. Le, "Bignas: Scaling up neural architecture search with big single-stage models," in European Conference on Computer Vision. Springer, 2020, pp. 702–717.

[29] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 2820–2828.

[30] H. Mao, M. Schwarzkopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh, "Learning Scheduling Algorithms for Data Processing Clusters," in ACM Special Interest Group on Data Communication, 2019, pp. 270–288.

[31] W.-L. Chiang et al., "Cluster-gcn: An Efficient Algorithm for Training Deep and Large Graph Convolutional Networks," in Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 257–266.

[32] Y. Xu, L. Xie, X. Zhang, X. Chen, G.-J. Qi, Q. Tian, and H. Xiong, "Pc-darts: Partial channel connections for memory-efficient architecture search," arXiv preprint arXiv:1907.05737, 2019.

[33] X. Dong and Y. Yang, "Searching for a robust neural architecture in four gpu hours," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 1761–1770.

[34] A. Zela, T. Elsken, T. Saikia, Y. Marrakchi, T. Brox, and F. Hutter, "Understanding and robustifying differentiable architecture search," arXiv preprint arXiv:1909.09656, 2019.

[35] X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive differentiable architecture search: Bridging the depth gap between search and evaluation," in Proceedings of the IEEE/CVF international conference on computer vision, 2019, pp. 1294–1303.

[36] H. Cai, L. Zhu, and S. Han, "ProxylessNAS: Direct neural architecture search on target task and hardware," in International Conference on Learning Representations, 2019.

[37] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-all: Train one network and specialize it for efficient deployment," in International Conference on Learning Representations, 2020.

[38] Paper with code, "Neural architecture search on imagenet." https://paperswithcode.com/sota/neural-architecture-search-on-imagenet, 2023.

[39] D. Stamoulis et al., "Single-Path NAS: Designing Hardware-Efficient ConvNets in less than 4 Hours," arXiv preprint arXiv:1904.02877, 2019.

[40] X. Chu, B. Zhang, and R. Xu, "Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search," in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 12 239–12 248.

[41] Z. Guo, X. Zhang, H. Mu, W. Heng, Z. Liu, Y. Wei, and J. Sun, "Single path one-shot neural architecture search with uniform sampling," in European conference on computer vision. Springer, 2020, pp. 544–560.

[42] W. Chen, X. Gong, X. Liu, Q. Zhang, Y. Li, and Z. Wang, "Fasterseg: Searching for faster real-time semantic segmentation," in International Conference on Learning Representations, 2020.

[43] M. Wu, H. Lin, and C. Tsai, "A training-free genetic neural architecture search," in ACM ICEA '21: 2021 ACM International Conference on Intelligent Computing and its Emerging Applications, Jinan, China, December 28 - 29, 2022. ACM, 2021, pp. 65–70.

[44] Y. Shu, Z. Dai, Z. Wu, and B. K. H. Low, "Unifying and boosting gradient-based training-free neural architecture search," CoRR, vol. abs/2201.09785, 2022.

[45] M. Javaheripi, S. Shah, S. Mukherjee, T. L. Religa, C. C. T. Mendes, G. H. de Rosa, S. Bubeck, F. Koushanfar, and D. Dey, "Litetransformersearch: Training-free on-device search for efficient autoregressive language models," CoRR, vol. abs/2203.02094, 2022.

[46] Q. Zhou, K. Sheng, X. Zheng, K. Li, X. Sun, Y. Tian, J. Chen, and R. Ji, "Training-free transformer architecture search," CoRR, vol. abs/2203.12217, 2022.

[47] T. M. Ingolfsson, M. Vero, X. Wang, L. Lamberti, L. Benini, and M. Spallanzani, "Reducing neural architecture search spaces with training-free statistics and computational graph clustering," in CF '22: 19th ACM International Conference on Computing Frontiers, Turin, Italy, May 17 - 22, 2022. ACM, 2022, pp. 213–214.

[48] L. T. Tran and S.-H. Bae, "Training-free hardware-aware neural architecture search with reinforcement learning," Journal of Broadcast Engineering, vol. 26, no. 7, pp. 855–861, 2021.

[49] L.-T. Tran, M. S. Ali, and S.-H. Bae, "A feature fusion based indicator for training-free neural architecture search," IEEE Access, vol. 9, pp. 133 914–133 923, 2021.

[50] T. Do and N. H. Luong, "Training-free multi-objective evolutionary neural architecture search via neural tangent kernel and number of linear regions," in International Conference on Neural Information Processing. Springer, 2021, pp. 335–347.

[51] L. Xiang, Ł. Dudziak, M. S. Abdelfattah, T. Chau, N. D. Lane, and H. Wen, "Zero-cost proxies meet differentiable architecture search," arXiv preprint arXiv:2106.06799, 2021.

[52] D. Zhou, X. Zhou, W. Zhang, C. C. Loy, S. Yi, X. Zhang, and W. Ouyang, "Econas: Finding proxies for economical neural architecture search," in Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition, 2020, pp. 11 396–11 404.

[53] K. Bhardwaj, G. Li, and R. Marculescu, "How does topology influence gradient propagation and model performance of deep networks with densenet-type skip connections?" in IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021. Computer Vision Foundation / IEEE, 2021.

[54] W. Chen, X. Gong, and Z. Wang, "Neural architecture search on imagenet in four gpu hours: A theoretically inspired perspective," in International Conference on Learning Representations, 2021.

[55] M. S. Abdelfattah, A. Mehrotra, Ł. Dudziak, and N. D. Lane, "Zero-cost proxies for lightweight nas," in International Conference on Learning Representations, 2021.

[56] X. He, K. Zhao, and X. Chu, "Automl: A survey of the state-of-the-art," Knowledge-Based Systems, vol. 212, p. 106622, 2021.

[57] M. Wistuba, A. Rawat, and T. Pedapati, "A survey on neural architecture search," arXiv preprint arXiv:1905.01392, 2019.

[58] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, X. Chen, and X. Wang, "A comprehensive survey of neural architecture search: Challenges and solutions," ACM Computing Surveys (CSUR), 2021.

[59] Y. Liu, Y. Sun, B. Xue, M. Zhang, G. G. Yen, and K. C. Tan, "A survey on evolutionary neural architecture search," IEEE transactions on neural networks and learning systems, 2021.

[60] L. Xie, X. Chen, K. Bi, L. Wei, Y. Xu, L. Wang, Z. Chen, A. Xiao, J. Chang, X. Zhang, and Q. Tian, "Weight-sharing neural architecture search: A battle to shrink the optimization gap," ACM Comput. Surv., vol. 54, no. 9, oct 2021.

[61] H. Benmeziane, K. El Maghraoui, H. Ouarnoughi, S. Niar, M. Wistuba, and N. Wang, "Hardware-aware neural architecture search: Survey and taxonomy," in Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21. International Joint Conferences on Artificial Intelligence Organization, 8 2021, pp. 4322–4329, survey Track.

[62] C. White, M. Khodak, R. Tu, S. Shah, S. Bubeck, and D. Dey, "A deeper look at zero-cost proxies for lightweight nas," in ICLR Blog Track, 2022, https://iclr-blog-track.github.io/2022/03/25/zero-cost-proxies/. [Online]. Available: https://iclr-blog-track.github.io/2022/03/25/zero-cost-proxies/

[63] X. Ning, C. Tang, W. Li, Z. Zhou, S. Liang, H. Yang, and Y. Wang, "Evaluating efficient performance estimators of neural architectures," in Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual, 2021, pp. 12 265–12 277.

[64] C. White, A. Zela, R. Ru, Y. Liu, and F. Hutter, "How powerful are performance predictors in neural architecture search?" in Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual, 2021, pp. 28 454–28 469.

[65] W. Chen, X. Gong, Y. Wei, H. Shi, Z. Yan, Y. Yang, and Z. Wang, "Understanding and accelerating neural architecture search with training-free and theory-grounded metrics," CoRR, vol. abs/2108.11939, 2021.

[66] W. Chen, W. Huang, and Z. Wang, ""no free lunch" in neural architectures? a joint analysis of expressivity, convergence, and generalization," in AutoML Conference 2023, 2023.

[67] O. Sharir and A. Shashua, "On the expressive power of overlapping architectures of deep learning," in 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net, 2018.

[68] X. Hu, L. Chu, J. Pei, W. Liu, and J. Bian, "Model complexity of deep learning: a survey," Knowl. Inf. Syst., vol. 63, no. 10, pp. 2585–2619, 2021.

[69] V. Nagarajan and J. Z. Kolter, "Generalization in deep networks: The role of distance from initialization," CoRR, vol. abs/1901.01672, 2019.

[70] L. Wu, Z. Zhu, and W. E, "Towards understanding generalization of deep learning: Perspective of loss landscapes," CoRR, vol. abs/1706.10239, 2017.

[71] S. Lin, "Generalization and expressivity for deep nets," IEEE Trans. Neural Networks Learn. Syst., vol. 30, no. 5, pp. 1392–1406, 2019.

[72] L. Xiao, J. Pennington, and S. S. Schoenholz, "Disentangling trainability and generalization in deep neural networks," in Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event, ser. Proceedings of Machine Learning Research, vol. 119. PMLR, 2020, pp. 10 462–10 472.

[73] V. Nagarajan and J. Z. Kolter, "Uniform convergence may be unable to explain generalization in deep learning," in Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, 2019, pp. 11 611–11 622.

[74] Y. Xu and H. Zhang, "Convergence of deep convolutional neural networks," Neural Networks, vol. 153, pp. 553–563, 2022.

[75] N. Lee, T. Ajanthan, and P. Torr, "SNIP: SINGLE-SHOT NETWORK PRUNING BASED ON CONNECTION SENSITIVITY," in International Conference on Learning Representations, 2019.

[76] H. Tanaka, D. Kunin, D. L. Yamins, and S. Ganguli, "Pruning neural networks without any data by iteratively conserving synaptic flow," in Advances in Neural Information Processing Systems, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 6377–6389.

[77] C. Wang, G. Zhang, and R. B. Grosse, "Picking winning tickets before training by preserving gradient flow," in International Conference on Learning Representations. OpenReview.net.

[78] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, "Importance estimation for neural network pruning," in IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019. Computer Vision Foundation / IEEE, 2019, pp. 11 264–11 272.

[79] Z. Zhang and Z. Jia, "Gradsign: Model performance inference with theoretical insights," in International Conference on Learning Representations, 2022.

[80] L. Theis, I. Korshunova, A. Tejani, and F. Huszár, "Faster gaze prediction with dense networks and fisher pruning," CoRR, vol. abs/1801.05787, 2018.

[81] L. Liu, S. Zhang, Z. Kuang, A. Zhou, J. Xue, X. Wang, Y. Chen, W. Yang, Q. Liao, and W. Zhang, "Group fisher pruning for practical network compression," in Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event, ser. Proceedings of Machine Learning Research, vol. 139. PMLR, 2021, pp. 7021–7032.

[82] V. Lopes, S. Alirezazadeh, and L. A. Alexandre, "Epe-nas: Efficient performance estimation without training for neural architecture search," in International Conference on Artificial Neural Networks. Springer, 2021, pp. 552–563.

[83] J. Mellor, J. Turner, A. Storkey, and E. J. Crowley, "Neural architecture search without training," in International Conference on Machine Learning. PMLR, 2021, pp. 7588–7598.

[84] M. Lin, P. Wang, Z. Sun, H. Chen, X. Sun, Q. Qian, H. Li, and R. Jin, "Zen-nas: A zero-shot nas for high-performance image recognition," in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 347–356.

[85] Z. Sun, M. Lin, X. Sun, Z. Tan, H. Li, and R. Jin, "MAE-DET: revisiting maximum entropy principle in zero-shot NAS for efficient object detection," in International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA, ser. Proceedings of Machine Learning Research, vol. 162. PMLR, 2022, pp. 20 810–20 826.

[86] L. Chizat, E. Oyallon, and F. R. Bach, "On lazy training in differentiable programming," in Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, 2019, pp. 2933–2943.

[87] A. Jacot, C. Hongler, and F. Gabriel, "Neural tangent kernel: Convergence and generalization in neural networks," in Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada, 2018.

[88] J. Lee, L. Xiao, S. S. Schoenholz, Y. Bahri, R. Novak, J. Sohl-Dickstein, and J. Pennington, "Wide neural networks of any depth evolve as linear models under gradient descent," in Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019,

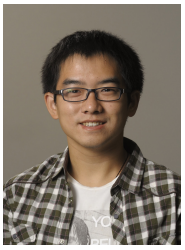
December 8-14, 2019, Vancouver, BC, Canada, 2019, pp. 8570–8581.
[89] W. Chen, W. Huang, X. Du, X. Song, Z. Wang, and D. Zhou, "Auto-scaling vision transformers without training," in International Conference on Learning Representations, 2022.
[90] M. Raghu, B. Poole, J. M. Kleinberg, S. Ganguli, and J. Sohl-Dickstein, "On the expressive power of deep neural networks," in Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017, ser. Proceedings of Machine Learning Research, vol. 70. PMLR, 2017, pp. 2847–2854.
[91] T. Serra, C. Tjandraatmadja, and S. Ramalingam, "Bounding and counting linear regions of deep neural networks," in Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018, ser. Proceedings of Machine Learning Research, vol. 80. PMLR, 2018, pp. 4565–4573.
[92] H. Xiong, L. Huang, M. Yu, L. Liu, F. Zhu, and L. Shao, "On the number of linear regions of convolutional neural networks," in Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event, ser. Proceedings of Machine Learning Research. PMLR, 2020.
[93] B. Hanin and D. Rolnick, "Complexity of linear regions in deep networks," in Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, ser. Proceedings of Machine Learning Research. PMLR, 2019.
[94] K. Bhardwaj, J. Ward, C. Tung, D. Gope, L. Meng, I. Fedorov, A. Chalfin, P. Whatmough, and D. Loh, "Restructurable activation networks," arXiv preprint arXiv:2208.08562, 2022.
[95] G. Li, S. K. Mandal, Ü. Y. Ogras, and R. Marculescu, "FLASH: fast neural architecture search with hardware optimization," ACM Trans. Embed. Comput. Syst., vol. 20, no. 5s, pp. 63:1–63:26, 2021.
[96] W. Chen, W. Huang, X. Gong, B. Hanin, and Z. Wang, "Deep architecture connectivity matters for its convergence: A fine-grained analysis," in Advances in Neural Information Processing Systems, 2022.
[97] J. Lee, J. Sohl-dickstein, J. Pennington, R. Novak, S. Schoenholz, and Y. Bahri, "Deep neural networks as gaussian processes," in International Conference on Learning Representations, 2018.
[98] J. Siems, L. Zimmer, A. Zela, J. Lukasik, M. Keuper, and F. Hutter, "Nas-bench-301 and the case for surrogate benchmarks for neural architecture search," arXiv preprint arXiv:2008.09777, 2020.
[99] Y. Mehta, C. White, A. Zela, A. Krishnakumar, G. Zabergja, S. Moradian, M. Safari, K. Yu, and F. Hutter, "Nas-bench-suite: Nas evaluation is (now) surprisingly easy," arXiv preprint arXiv:2201.13396, 2022.
[100] A. Mehrotra, A. G. C. Ramos, S. Bhattacharya, Ł. Dudziak, R. Vipperla, T. Chau, M. S. Abdelfattah, S. Ishtiaq, and N. D. Lane, "Nas-bench-asr: Reproducible neural architecture search for speech recognition," in International Conference on Learning Representations, 2020.
[101] N. Klyuchnikov, I. Trofimov, E. Artemova, M. Salnikov, M. Fedorov, A. Filippov, and E. Burnaev, "Nas-bench-nlp: neural architecture search benchmark for natural language processing," IEEE Access, vol. 10, pp. 45 736–45 747, 2022.
[102] C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter, "Nas-bench-101: Towards reproducible neural architecture search," in International Conference on Machine Learning. PMLR, 2019, pp. 7105–7114.
[103] X. Dong and Y. Yang, "NAS-Bench-201: Extending the Scope of Reproducible Neural Architecture Search," arXiv preprint arXiv:2001.00326, 2020.
[104] X. Dong, L. Liu, K. Musial, and B. Gabrys, "Nats-bench: Benchmarking nas algorithms for architecture topology and size," IEEE transactions on pattern analysis and machine intelligence, 2021.
[105] Y. Duan, X. Chen, H. Xu, Z. Chen, X. Liang, T. Zhang, and Z. Li, "Transnas-bench-101: Improving transferability and generalizability of cross-task neural architecture search," in IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021. Computer Vision Foundation / IEEE, 2021.
[106] C. Li, Z. Yu, Y. Fu, Y. Zhang, Y. Zhao, H. You, Q. Yu, Y. Wang, and Y. Lin, "Hw-nas-bench: Hardware-aware neural architecture search benchmark," arXiv preprint arXiv:2103.10584, 2021.
[107] L. Dudziak, T. Chau, M. S. Abdelfattah, R. Lee, H. Kim, and N. D. Lane, "BRP-NAS: prediction-based NAS using gcns," in Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, 2020.
[108] H. Lee, S. Lee, S. Chong, and S. J. Hwang, "Hardware-adaptive efficient latency prediction for nas via meta-learning," in Advances in Neural Information Processing Systems, 2021.
[109] L. L. Zhang, S. Han, J. Wei, N. Zheng, T. Cao, Y. Yang, and Y. Liu, "nn-meter: towards accurate latency prediction of deep-learning model inference on diverse edge devices," in MobiSys '21: The 19th Annual International Conference on Mobile Systems, Applications, and Services, Virtual Event, Wisconsin, USA, 24 June - 2 July, 2021. ACM, 2021, pp. 81–93.
[110] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," IEEE Trans. Image Process., vol. 13, no. 4, pp. 600–612, 2004.
[111] R. Wightman, "Pytorch image models," https://github.com/rwightman/pytorch-image-models, 2019.
[112] RangiLyu, "Nanodet-plus: Super fast and high accuracy lightweight anchor-free object detection model." https://github.com/RangiLyu/nanodet, 2021.
[113] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang, "The expressive power of neural networks: A view from the width," in Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, 2017, pp. 6231–6239.
[114] K. Hornik, M. B. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," Neural Networks, vol. 2, no. 5, pp. 359–366, 1989.
[115] N. Tripuraneni, B. Adlam, and J. Pennington, "Overparameterization improves robustness to covariate shift in high dimensions," Advances in Neural Information Processing Systems, 2021.
[116] Z. Allen-Zhu, Y. Li, and Y. Liang, "Learning and generalization in overparameterized neural networks, going beyond two layers," in Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, 2019, pp. 6155–6166.
[117] G. Li, Y. Yang, K. Bhardwaj, and R. Marculescu, "Zico: Zero-shot nas via inverse coefficient of variation on gradients," arXiv preprint arXiv:2301.11300, 2023.

Guihong Li (Student Member, IEEE) received the B.S degree from the Beijing University of Posts and Telecommunications, Bejing, China, in 2018. He is currently pursuing his Ph.D. in Electrical and Computer Engineering at The University of Texas at Austin, USA. His research interest includes Neural Architecture Search, hardware-software co-design for EdgeAI system optimization. He received many awards including a best paper nomination from ISWC 2022.

Duc Hoang recevied a Bachelor's degree in ECE from the University of Washington. He is currently pursuing a Ph.D. in the same field at the University of Texas at Austin, focusing on Graph Neural Networks, Neural Architectural Search, and Network Pruning.

Kartikeya Bhardwaj is a Senior Machine Learning Researcher at Qualcomm AI Research. Previously, he was a Senior Machine Learning Engineer at Arm, Inc. He completed his PhD in Electrical and Computer Engineering from Carnegie Mellon University in 2019. His research interests are in the field of hardware-aware deep learning, computer vision, and network science. His work has been published in top conferences including CVPR, ICLR, MLSys, ECML, DAC, DATE, etc.

Ming Lin is a Senior Applied Scientist at Amazon.com LCC. His research interests include Mathematical Foundation of Deep Learning and Statistical Machine Learning, with their applications in deep learning acceleration, computer vision and mobile AI. He worked as a postdoctoral researcher in the School of Computer Science at Carnegie Mellon University from July 2014 to Sep 2015. He received his Ph.D. degree in computer science from Tsinghua University in 2014. During his Ph.D. study, he had been a visiting scholar in Michigan State University and in CMU from Dec 2013 to July 2014.



Zhangyang Wang is currently the Temple Foundation Endowed Associate Professor #7 of ECE at UT Austin. He received his Ph.D. in ECE from UIUC in 2016, and his B.E. in EEIS from USTC in 2012. Prof. Wang is broadly interested in the fields of machine learning, computer vision, optimization, and their interdisciplinary applications. His latest interests focus on the role of low dimensionality in deep learning.

Radu Marculescu is the Laura Jennings Turner Chair in Engineering and Professor in the Electrical and Computer Engineering department at The University of Texas at Austin. He received his Ph.D. in Electrical Engineering from the University of Southern California in 1998. Radu's current research focuses on developing ML/AI methods and tools for modeling and optimization of embedded systems, cyber-physical systems, and social networks.