Extensible and Efficient Proxy for Neural Architecture Search

Yuhong Li[1], Jiajie Li[2], Cong Hao[3], Pan Li[3], Jinjun Xiong[2], Deming Chen[1]
University of Illinois at Urbana-Champaign[1], University at Buffalo[2]
Georgia Institute of Technology[3],
[1]{leeyh,dchen}@illinois.edu [2]{jli433,jinjun}@buffalo.edu [3]{callie.hao,panli}@gatech.edu

Abstract

Efficient or near-zero-cost proxies were proposed recently to address the demanding computational issues of Neural Architecture Search (NAS) in designing deep neural networks (DNNs), where each candidate architecture network only requires one iteration of backpropagation. The values obtained from proxies are used as predictions of architecture performance for downstream tasks. However, two significant drawbacks hinder the wide adoption of these efficient proxies: (1) they are not adaptive to various NAS search spaces*; and (2)* they are not extensible to multi-modality downstream tasks*. To address these two issues, we first propose an Extensible proxy (Eproxy) that utilizes self-supervised, few-shot training to achieve near-zero costs. A key component to our Eproxy's efficiency is the introduction of a* barrier layer *with randomly initialized frozen convolution parameters, which adds non-linearities to the optimization spaces so that Eproxy can discriminate the performance of architectures at an early stage. We further propose a Discrete Proxy Search (DPS) method to find the optimized training settings for Eproxy with only a handful of benchmarked architectures on the target tasks. Our extensive experiments confirm the effectiveness of both Eproxy and DPS. On the NDS-ImageNet search spaces, Eproxy+DPS achieves a higher average ranking correlation (Spearman ρ = 0.73) than the previous efficient proxy (Spearman ρ = 0.56). On the NAS-Bench-Trans-Micro search spaces with seven tasks, Eproxy+DPS delivers comparable performance with the early stopping method (146× faster). For the end-to-end task such as DARTS-ImageNet-1k, our method delivers better results than NAS performed on CIFAR-10 while only requiring one GPU hour with a single batch of CIFAR-10 images. Our code is available at https://github.com/leeyeehoo/GenNAS-Zero.*

1. Introduction

As deep neural networks (DNNs) find uses in a wide range of applications, such as computer vision [23, 27, 46, 48] and natural language processing [11,24,47,50,58], Neural Architecture Search (NAS) [4,37,45,49,66] has become an increasingly important technique to automate the design of neural architectures for different tasks [21, 38, 53, 54]. Recent progress in NAS has demonstrated superior results, surpassing those of human designs [49, 56, 66]. However, one major hurdle for NAS is its high computation cost. For example, the seminal work of NAS [66] consumed 2000 GPU hours to obtain a high-quality DNN, a prohibitively high cost for many researchers. The high computation cost of NAS can be attributed to three major factors: (1) the large search space for candidate neural architectures, (2) the training of various candidate neural architectures, and (3) the comparison of the solution quality of candidate neural architectures to guide the NAS search process. Subsequent NAS work has proposed various techniques to address the above issues, such as the constraints on the search space [30], the weight-sharing networks to reduce the training cost [43], and search with synthetic or random labels [35, 65].

Out of the advancement, the latest efficient proxies showed that the quality of a neural architecture could be determined by a proxy metric computed within seconds without full training. Hence they are near zero-cost (ZC). The activations of an untrained network were analyzed as a proxy in NASWOT [41] with promising results. Abdelfattah et al. [1] suggested various proxies for pruning. Zhang et al. [64] proposed a zero-cost pruning method for differentiable NAS at initialization. ZenNAS [34] quantifies network expressivity, which has a positive correlation with model accuracy. ZiCo [31] leverages gradient properties to improve neural network convergence rate. The efficient proxies discussed above have, however, two primary drawbacks:

1. The quality of these efficient proxies is inconsistent across different search spaces. While many proxies show high correlations in the confined search spaces of small NAS Benchmarks, their performance can be vastly different in real-world applications where the search spaces are orders of magnitude larger than those of the tabular benchmarks. For instance, Synflow demonstrates a high ranking correlation on NAS-

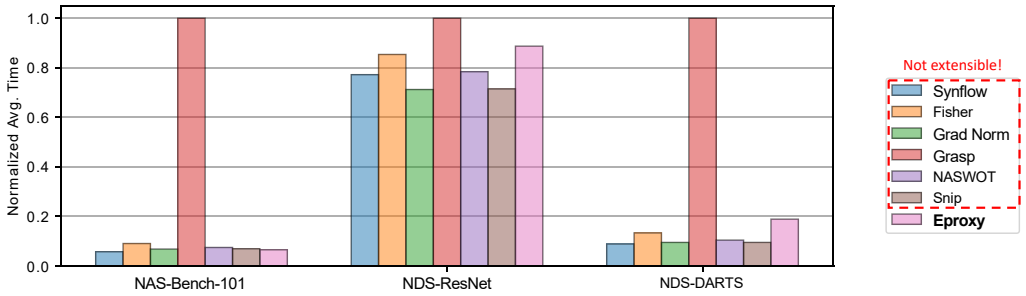


Figure 1. Eproxy is compared to six efficient proxies in terms of evaluation speed on NAS-Bench-101, NDS-ResNet, and NDS-DARTS search spaces, demonstrating that Eproxy falls within the zero-cost category. The plot shows the normalized average time.

Bench-201 [16], but its performance is subpar on NAS-Bench-101 [62], which is 27 times larger than NAS-Bench-201.

2. Efficient proxies don't adapt well to multi-modality downstream tasks. These proxies are typically tailored for CIFAR-10 like classification tasks, resulting in promising prediction results for these specific tasks. However, their performance drops significantly in other scenarios. As an illustration, NASWOT exhibits a dismal average ranking correlation of 0.03 on NAS-Bench-MR [13], which consists of nine real-world tasks. Many efficient proxies employ specific algorithms, like pruning, to convert the weights of architectures into prediction values. This rigid algorithmic approach hinders the proxy's adaptability to tasks beyond classification. Additionally, some zero-cost proxies have been found to favor certain neural architectures [6]. For example, both empirical and theoretical evidence suggest that Synflow has a preference for large models [42].

We question if it is possible for a zero-cost, few-shot proxy task to accurately mimic the true task and result in similar performance ranks for architectures, as shown in Fig. 2. This leads us to introduce a novel and efficient proxy, called the Extensible proxy (Eproxy), approached from a different perspective. Recently, in our prior work, Li et al. [33] introduced a method that employs regression-based training termed GenNAS for the assessment of architectures. Although innovative, their approach is not devoid of costs, both in terms of time and resources. In contrast, our work takes inspiration from the GenNAS framework, achieving nearly zero-cost evaluations. Unlike previous efficient proxies, Eproxy utilizes few-shot spatial-level regression on a set of image-label pairs (see Illustration in Fig. 4). The labels are 2D synthetic features because, as suggested by Li et al. [33], spatial-level regression is more challenging than one-hot classification on a tiny dataset, i.e., a batch of image-label pairs. The key component of Eproxy is the barrier layer. It takes the output of the architecture network into an untrained convolutional layer and performs the regression with the labels. Such a simple mechanism can significantly improve the performance of Eproxy to identify good architectures and bad ones when performing 10 iterations of backpropagation, i.e., near zero-cost. We find that the barrier layer increases the complexity of the optimization space. Hence, poor-performance architectures are more difficult to optimize (see Section 3.1). Since Eproxy is a configurable few-shot trainer, we design a novel search space for Eproxy that includes various hyperparameters, such as feature combinations, output channel numbers, and selection for barrier layers, which makes up Eproxy's multi-modalities. We term such a search method as Discrete Proxy Search (DPS) (the performance of which is shown in Fig. 3). Notably, besides the evaluation of a handful of architectures' performance, DPS does not need to use any task-specific information (e.g., in our experiment, we only use a single batch of CIFAR-10 [26] images throughout all the experiments).

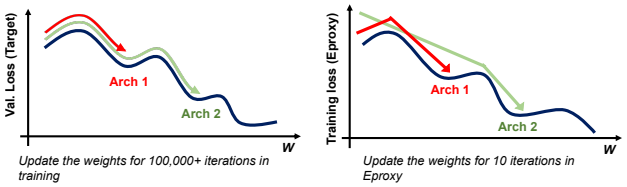
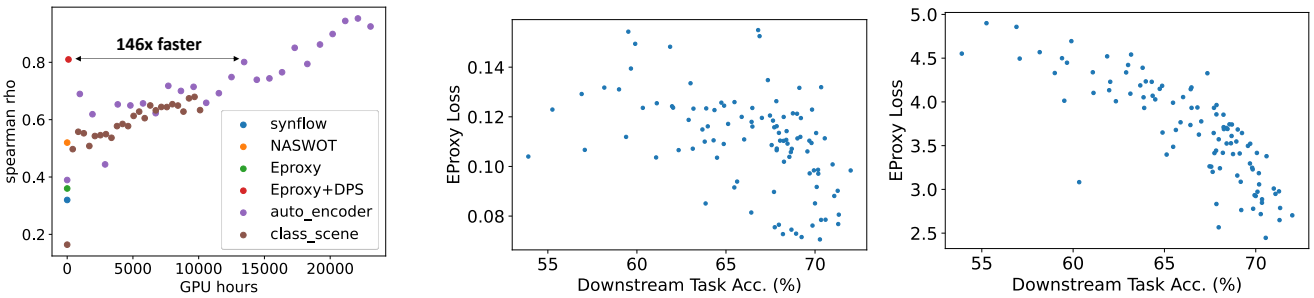


Figure 2. A hypothetical illustration of the validation losses of two architectures on a downstream task is shown on the left. The losses of a sophisticated few-shot proxy, on the right, can reflect the actual performance of the architectures on target tasks. The "W" represents updated weights.

We summarize our contributions as follows:

1. We propose an efficient proxy task with the barrier layer that utilizes a few-shot self-supervised regression. The task adopts one batch of images in the CIFAR-10-level dataset (not necessarily from the target training dataset). It uses synthetic labels to evaluate architectures. Eproxy significantly speeds up the traditional early stopping evaluation process while main-



(a) Autoencoder: Time vs Corr. (b) DARTS-ImgNet: Eproxy (c) DARTS-ImgNet: Eproxy+DPS

Figure 3. **a:** Comparison with efficient proxies and early stopping methods on NAS-Bench-Trans-Micro Autoencoder task. It shows the effectiveness of DPS compared with early stopping methods on either the target task or a classification task when evaluating 4096 architectures. **b, c:** On NDS DARTS-**ImageNet** task, Eproxy and Eproxy+DPS (Searched on DARTS-CIFAR-10, transferred to ImageNet) achieve 0.51, 0.85 ρ respectively. It shows DPS can find a search-space-aware Eproxy.

taining the high-ranking correlation.

2. We propose the downstream-task and search-space aware proxy search algorithm with a proxy search space. We formulate the proxy task search as a discrete optimization problem with only a handful of architectures, such that the performance rankings of the networks on the ground-truth task and the proxy task should be consistent. The searched Eproxy can accurately evaluate the quality of network architectures and make Eproxy search-space and downstream-task aware.

3. We provide thorough experiments to evaluate the performance of Eproxy and Eproxy boosted by DPS on more than 30 search spaces and tasks. We demonstrate that our methods have overall higher performance than existing efficient proxies in terms of all three factors: architecture ranking correlation score, top-10%-architecture retrieve rate, and end-to-end NAS performance. Our solid experimental results can be further utilized to benefit the NAS community.

2. Our Approaches

In Sec.2.1, we present the Eproxy for efficient network evaluation. Sec.2.2 explains how to find a task and search-space aware Eproxy using Discrete Proxy Search.

2.1. Extensible NAS Proxy

The Eproxy is designed for the architectures to learn the output of an untrained network on a set of image-label pairs (See Fig. 4). We utilize the MSE-based training [33] with a large learning rate and limited backpropagation steps to make it as efficient as the existing near zero-cost proxies. However, directly applying a few-shot regression task with a large learning rate leads to poor correlation based on our observations. To make the Eproxy architecture-performance-aware within a few iterations, we propose an untrained barrier layer to make the task more involved (See Section 3.1). The barrier layer is a randomly initialized convolution layer to the output of the trainable components. Our experiments show that adding such a layer can significantly improve the correlation between the predictions and the performance of neural architectures in the downstream tasks within a few back propagations (Sec 3.1). To be more specific, the Eproxy training loss can be described as:

$$\min_{w_a, w_t} \mathcal{L}_{MSE}(G(w_b, F(w_a, w_t, X)), Y) \tag{1}$$

where the $X \in \mathbb{R}^{b \times c_{in} \times h_{in} \times w_{in}}$ is the a set of input images (b is batch size; c_{in} is number of input channels). F is a fully convolutional neural network (FCN) with a transform layer (a convolutional layer) that transforms the X to $F(\cdot) \in \mathbb{R}^{b \times c_{mid} \times h_{out} \times w_{out}}$. The FCN is obtained using the architecture without a task-specified head in the downstream tasks. For example, the classifier network with the classification (last pooling and linear layer) head was removed. w_a and w_t are the weights of architecture for evaluation and the weights of the transform layer that project the output channels of the architecture to c_{mid} which is the number of the transform layer's output channels. G is the barrier layer, and w_b is the weights. Note in the Eproxy without DPS, $Y \in \mathbb{R}^{b \times c_{out} \times h_{out} \times w_{out}}$ is the output of an untrained 6-layer CNN (Fig. 4, 'Net').

We further provide the pseudo code of Eproxy in Listing 1. The Eproxy utilizes a batch of images and corresponding synthetic features as labels (configurations such as the combination of features and output channel numbers can be searched by DPS). The model is trained for 10 iterations. The model's performance is gauged by the MSE loss (lower values denoting better performance).

```
def Eproxy(arch, barrier, img, features,
    t_iter = 10, **kwargs):
    # img shape: B, C = 3, W_in, H_in
    # features shape: B,  C_out, W_out,
    H_out
```

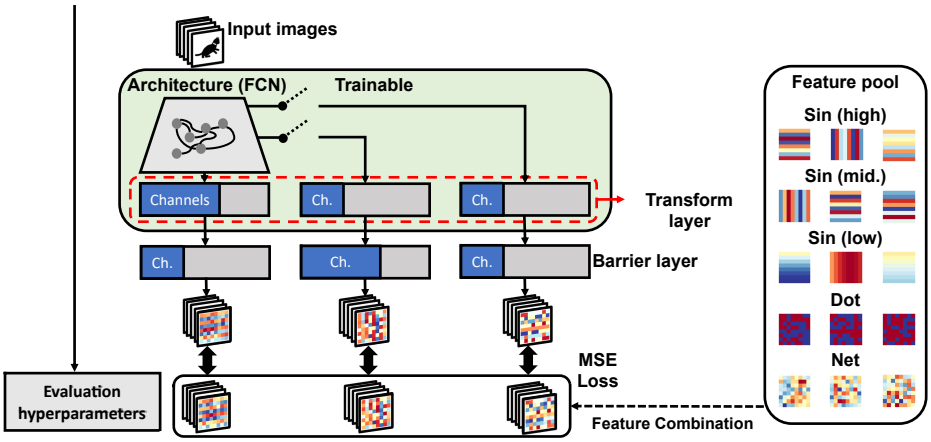


Figure 4. The design of Eproxy and the searchable components. Dotted line: The configurable components for Discrete Proxy Search. Green block: Trainable components. DPS can further utilize the configuration of Eproxy to target search spaces and downstream tasks.

```
    optimizer = SGD(arch.parameters())
    for i in range(t_iter):
        output_mid = model(img) # B, C_mid,
     W_out, H_out
        output = barrier(output_mid) # B,
    C_out, W_out, H_out
        loss = ((output - features)**2).
    mean()
        optimizer.zero_grad()
        loss_m.backward()
        optimizer.step()
    return loss
```

Listing 1. Pseudo PyTorch-style code for Eproxy.

2.2. Discrete Proxy Search

Since Eproxy provides abundant configurable hyperparameters and utilizes data-agnostic spatial labels, different settings can be naturally adjusted for tasks and search spaces. Therefore, we propose a semi-supervised discrete proxy search to find a setting that can be suitable for the specific modality. As shown in Fig. 4, the searchable configurations are provided as follows:

1. Transform and barrier layer: Both layers can have kernel size selected from $\{1, 3, 7\}$, and the channel number c_{mid} can be selected from 16 to 512 geometrically with 2 as a multiplier.

2. Feature combination: a) Untrained CNN outputs. The experiment results show that an untrained network's output features can be powerful for evaluating architectures on numerous tasks/search spaces. The synthetic features can be interpreted as a tiny knowledge-distillation task from an untrained teacher network. b) Sine wave: we adopt the sine wave features with low/mid/high frequency along width/height. The insight is that good CNNs can learn different frequency signal [33, 61]. The features are generated by $\sin(2\pi fx + \phi)$ or $\sin(2\pi fy + \phi)$ with equal probability to be selected. We set three ranges for frequency f: low (L) $f \in (0, 0.125)$, medium (M) $f \in (0.125, 0.375)$, and high (H) $f \in (0.375, 0.5)$. c) Dot: By utilizing the Rademacher distribution, we generate the synthetic features with only ± 1. The features attempt to simulate the spatial classification that is widely adopted in tasks such as detection [19], segmentation [3], tracking [3, 29]. The combined features can be multiplied by an augment coefficient selected from 0.5 to 2 with 0.5 as a step.

3. Training hyperparameters: a) Learning rate: we adopt the SGD optimizer, and the learning rate can be selected from 0.5 to 1.5 with 0.1 as the step. b) Initialization: we adopt two initialization methods, Kaiming [22] and Xavier [20], with either Gaussian or Uniform initialization, resulting a total of four choices.

4. Intermediate output evaluation: We provide the choices to force the network to learn the intermediate outputs from the layer before the first or second downsample layer. The motivation is that earlier stages of the network have different learning behaviors from the deeper stages [2]. Thus, monitoring the early stages can give more flexibility for adapting Eproxy to different tasks.

5. FLOPS: As works [25, 42, 55, 56] suggested that FLOPS is a good indicator for architecture performance. Hence we incorporate the FLOPS normalized by the largest architecture in the search space with the Eproxy loss as $\mathcal{L} \cdot (1 + \alpha \cdot \text{norm(FLOPS)})$, where α can be selected from -0.5 to 0.5 with 0.1 steps.

The total number of configuration combinations in the proxy search space is 5×10^{15}. We utilize the regularization evolutionary algorithm (REA) [45] to conduct the exploration. First, we randomly sample a small subset of the neural architectures in the NAS search space and obtain their ground truth ranking on the target task or a highly correlated down-scaled task (for example, CIFAR-10 is a good proxy for ImageNet). We then evaluate these networks using Eproxy with different configurations and calculate the performance ranking correlation ρ of the Eproxy and the target task, and the ρ is the fitness function for REA.

We provide the pseudo code of the DPS function in Listing 2. This function utilizes a given set of architectures and their corresponding ground-truth accuracies (`archs_accs`) and iterates over a specified number of cycles (`cycle`). DPS starts with an initial population of configurations, each of which includes parameters like learning rate, channel numbers, feature combinations, among others. The `REAEngine` class is utilized to manage the Regularized Evolutionary Algorithm. It aids in generating random configurations, calculating correlations between Eproxy losses (under the given configuration) and ground-truth accuracies for a set of architectures, and performing evolutionary mutations on the configurations. The function accumulates and returns the history of configurations and their corresponding correlations over the evolutionary cycles in `config_history`.

```
def DPS(archs_accs, cycle, population = 40,
     sample = 10, **kwargs):
    # len(archs_accs): 20 pairs of archs &
    gt accs.
    # config: including lr, channel number,
     feature combination, etc.
    config_history = []
    rea = REAEngine(population, sample,
    mutation_rate)
    # generate the initial pool
    for _ in range(population):
        config = rea.get_random_config()
        corr = rea.get_corr(config,
    archs_accs) # calculate the correlation
    between Eproxy (under the config) losses
     and gt accs based on 20 archs.
        config_history.append({config: corr
    })
    # evolution
    for _ in range(cycle):
        new_config = rea.get_mutate_config
    ()
        corr = rea.get_corr(new_config,
    archs_accs)
        config_history.append({new_config:
    corr})
    return config_history
```

Listing 2. Pseudo PyTorch-sytle code for DPS.

3. Experiments

In this section, we perform the following evaluations for Eproxy and DPS. First, in Sec. 3.1, we conduct the ablation study on NASBench-101 [62], the first and yet the largest tabular NAS benchmark with over 423k CNN models and training statistics on CIFAR-10. We explain the mechanism behind the barrier layer with empirical results. Furthermore, we compared Eproxy and Eproxy boosted by DPS with existing efficient proxies. Second, from Sec. 3.2 to Sec. 3.4, we use metrics including ranking correlation, top-10 architecture retrieve rate [12] to evaluate the proposed method on **NDS** [44] (11 search spaces on CIFAR-10, 8 search spaces on ImageNet), **NAS-Bench-Trans-Micro** [17] (7 tasks), and **NAS-Bench-MR** [13] (9 tasks). Third, in Sec. 3.5, we evaluate the end-to-end NAS on NAS-Bench-101/201. Moreover, we report the end-to-end search on the DARTS-ImageNet search space in Sec. 3.5.

Loss	MSE w/o Barrier			MSE w/ Barrier		
LR	1	1e-1	1e-2	1	1e-1	1e-2
10 iters^{NZC}	0.08	-0.22	-0.19	**0.65**	0.46	0.09
100 iters	0.07	0.67	0.76	0.65	**0.79**	0.79
200 iters	0.22	0.64	0.66	0.61	**0.83**	0.81

Table 1. Ranking correlation (Spearman's ρ) analysis for different losses on NASBench-101. "LR" stands for learning rate; "NZC" stands for near-zero-cost. The results suggest that regression with barrier and large learning rate can achieve a high ranking correlation in 10 iterations near zero cost.

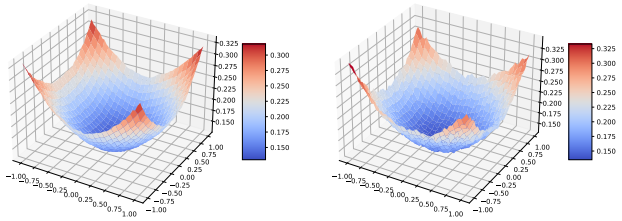
3.1. Ablation Study on NAS-Bench-101

We study the effectiveness of our barrier layer in this section. We use the tool from the work [32] to visualize the loss surface of an architecture selected randomly from NAS-Bench-101 on our few-shot regression task. Figure 5a,5b shows the loss surface without the barrier has a good convexity, which indicates the task is simple, as we use a proxy task that contains very few samples (16 image-label pairs) for a shorter evaluation period. The simplicity of the proxy task gives us two potential problems that can affect the final results. (1) If a task is too simple, every model can perform similarly well. (2) When the optimization is easy, models can have similar performance at the early stage of training. As we observed, loss surfaces from different models have similar shapes without barriers, requiring us to use more training steps to see the difference between good and bad architectures. To mitigate these two problems, Eproxy added a barrier layer which is a random initialized linear/convolution layer with frozen weights. As shown in Figure 4 (b), the loss surface with the barrier has a noticeable non-convexity, which shows the increased complexity of the proxy task, and now it can better reflect the actual performance of ar-

	Grad norm	Snip	Grasp	Fisher	Synflow	NASWOT	ZenNAS	ZiCo	Eproxy	Eproxy+DPS
ρ	0.20	0.16	0.45	0.26	0.37	0.40	0.61	0.61	0.65	**0.69**
Top-10%	2%	3%	26%	3%	23%	29%	39%	**45%**	31%	38%

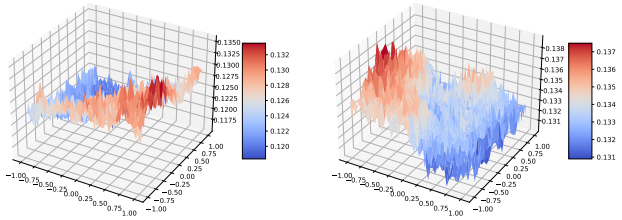
Table 2. Comparison with efficient proxies on NAS-Bench-101 using the Spearman ρ and top-10% retrieve rate. Eproxy outperforms the state-of-the-art zero-shot methods, ZiCo and ZenNAS, in terms of rho when DPS is not utilized.

chitecture (Figure 6a,6b). As the irregular shape of the loss surface varies widely from model to model, it helps us better distinguish the model performance at the early stage of training, allowing us to use fewer training steps to speed up the evaluation further. The results in Table 1 show that with the barrier layer, Eproxy can reach ρ 0.65 in only 10 iterations with a learning rate of 1, and it also significantly improves the ranking correlation score with more training iterations.



(a) Best model without barrier. (b) Worst model without barrier.

Figure 5. The loss surfaces of best and worst model from NAS-Bench-101 regression task without barrier.



(a) Best model with the barrier. (b) Worst model with the barrier.

Figure 6. The loss surfaces of best and worst model from NAS-Bench-101 regression task with barrier.

Next, we sample 20 architectures from NAS-Bench-101 and evaluate DPS. We conduct DPS for 200 epochs, and the total run time is ∼20 mins on a single A6000 GPU. In Table 2, we report the network evaluation results in terms of Spearman's ρ and top-10% network coverage using the proxy task searched by DPS. We evaluated Eproxy against state-of-the-art zero-cost proxies including NASWOT [41], Synflow [1], ZenNAS [34], and ZiCo [31]. Additionally, we considered architecture parameters and FLOPS as important baselines [55]. Eproxy significantly outperforms existing zero-cost proxies by a large margin. For instance, Synflow, scores 0.45, ZiCo scores 0.61, Eproxy scores 0.65 (without DPS), and Eproxy with DPS scores 0.69. In terms of top-10% retrieval rate, Eproxy with DPS retrieves a higher number of architectures than DPS (38% vs 31%). The results support the efficiency and effectiveness of DPS. Meanwhile, Fig. 1 confirms that using Eproxy achieves a comparable evaluation speed with other efficient proxies.

3.2. NDS

The original Network Design Spaces (NDS) work [44] investigates different search spaces. The NDS is perfect for evaluating efficient proxies in more complex search spaces. For example, researchers benchmarked 5,000 architectures on DARTS search space and over 20,000 on ResNet search space. We compared our method with existing zero-cost proxies on **11 search spaces** on **CIFAR-10** and **8 search spaces** on **ImageNet** [10]. We show the results in Table 3. Compared to Synflow, ZenNAS, and ZiCo, Eproxy (without DPS) delivers better results on both CIFAR-10 and ImageNet search spaces. Eproxy (without DPS) performs similarly to NASWOT on both CIFAR-10 and ImageNet search spaces. Boosted by DPS, Eproxy delivers significantly better results on target CIFAR-10 search spaces with 36% and 52% improvement on ranking correlation and top-10% retrieve rate, respectively. Notably, Eproxy+DPS searched on CIFAR-10 with 20 architectures performs significantly better on **ImageNet** search spaces without any prior knowledge of the dataset. Compared to NWT, Eproxy+DPS gains 30% and 57% on ranking correlation and top-10% retrieve rate, respectively. The ImageNet experiment demonstrates the efficiency by utilizing the architectures trained on down-scaled dataset (CIFAR-10) for DPS. It's noteworthy that using FLOPS and Params as evaluation metrics for models may not be suitable in search spaces with limited model size variations, such as DARTS-f, PNAS-f, and ENAS-f.

3.3. NAS-Bench-Trans-Micro

Previous experiments suggest that DPS can optimize Eproxy across different search spaces. We further evaluate Eproxy and DPS on NAS-Bench-Trans-Micro, a benchmark that contains 4096 architectures across **7 large tasks** from the **Taskonomy** [63] dataset. The tasks include object classification, scene classification, unscrambling the image, and image upscaling. The search space is similar to NAS-Bench-201 but has 4 operator choices per edge instead of

CIFAR-10	DARTS	DARTS-f	AMB	ENAS	ENAS-f	NASNet	PNAS	PNAS-f	Res	ResX-A	ResX-B	Avg.
Synflow	0.42 9%	-0.14 5%	-0.10 3%	0.18 6%	-0.30 2%	0.02 7%	0.25 9%	-0.26 4%	0.21 4%	0.47 25%	0.61 29%	0.12 9%
NASWOT	0.65 29%	0.31 8%	0.29 20%	0.54 31%	0.44 28%	0.42 27%	0.50 24%	0.13 6%	0.29 7%	0.64 28%	0.57 21%	0.43 21%
ZenNAS	0.50 14%	0.01 4%	0.05 7%	0.22 10%	0.07 5%	0.07 11%	0.24 12%	0.20 3%	0.23 2%	0.59 35%	**0.66** 32%	0.26 12%
ZiCo	0.49 13%	0.11 5%	0.09 4%	0.29 12%	0.02 11%	0.16 12%	0.26 9%	0.09 3%	0.23 3%	0.54 32%	0.63 **34%**	0.26 13%
Eproxy	0.38 12%	0.34 17%	0.54 13%	0.59 35%	0.48 31%	0.56 28%	0.22 4%	0.24 4%	0.51 36%	0.47 24%	0.19 10%	0.41 19%
Eproxy+DPS	**0.72** 33%	**0.39** **19%**	**0.56** **29%**	**0.63** **36%**	**0.47** **30%**	**0.54** **32%**	**0.60** **35%**	**0.48** **28%**	**0.56** **36%**	**0.65** **32%**	0.60 19%	**0.56** **29%**

ImageNet	DARTS	DARTS-f	Amoeba	ENAS	NASNet	PNAS	ResX-A	ResX-B	Avg.
Synflow	0.21 0%	-0.36 4%	-0.25 0%	0.17 9%	0.01 0%	0.14 9%	0.42 7%	0.31 13%	0.08 6%
NASWOT	0.66 16%	0.20 8%	0.42 33%	0.69 36%	0.51 33%	**0.61** 10%	0.73 30%	0.63 38%	0.56 26%
ZenNAS	0.21 8%	0.13 4%	0.21 0%	0.22 18%	0.06 17%	0.23 9%	0.60 23%	0.45 25%	0.27 13%
ZiCo	0.24 8%	0.01 4%	0.18 0%	0.26 36%	0.12 17%	0.27 9%	0.52 8%	0.40 19%	0.25 13%
Eproxy	0.51 20%	0.31 17%	0.66 60%	0.58 33%	0.56 30%	0.36 33%	0.73 55%	0.70 **43%**	0.55 36%
Eproxy+DPS$_{\mathrm{T}}$	**0.85** 50%	**0.53** **28%**	**0.66** **60%**	**0.79** 33%	**0.85** 32%	0.60 **35%**	**0.83** **55%**	**0.72** 36%	**0.73** **41%**

Table 3. Comparison with efficient proxies on NDS search spaces. $_{\mathrm{T}}$ denotes the DPS on CIFAR-10 and transferred to ImageNet. Therefore, it does not necessitate knowledge of the ImageNet dataset. When DPS is not employed, Eproxy performs better than ZiCo on 7 out of 11 CIFAR-10 search spaces and all 8 ImageNet search spaces. However, with DPS, Eproxy shows significant superiority over other methods.

	Cls. Scene	Cls Obj	Room Layout	Jigsaw	Seg	Normal	AE	Avg.
Synflow	0.46/16%	0.50/16%	0.45/28%	0.49/19%	0.32/3%	0.52/19%	0.52/34%	0.47/19%
NASWOT	0.57/21%	0.53/21%	0.30/2%	0.41/11%	0.52/30%	0.59/30%	-0.02/2%	0.41/17%
ZenNAS	0.57/**48%**	0.34/32%	0.24/**27%**	0.35/35%	0.37/32%	0.56/47%	0.15/23%	0.37/35%
ZiCo	0.30/31%	0.03/16%	0.04/23%	0.15/26%	0.07/16%	0.29/27%	0.19/9%	0.15/21%
Eproxy	0.15/14%	0.45/34%	0.06/8%	0.17/33%	0.36/46%	0.25/38%	0.61/**80%**	0.29/36%
Eproxy + DPS	0.70/30%	0.56/**44%**	**0.56**/13%	0.64/**45%**	**0.81/53%**	**0.81/63%**	**0.80**/74%	**0.69/46%**
ES$_{\sim 660\text{GPU hrs/task}}$	**0.73**/25%	0.01/7%	0.15/7%	**0.74**/21%	0.39/7%	0.65/27%	0.35/11%	0.43/15%

Table 4. Comparison with efficient proxies and the early stopping method on TransNAS-Bench-Micro. In 5 out of 7 tasks, Eproxy surpasses ZiCo, a recent zero-shot method. Eproxy+DPS outperforms efficient proxies, and the early stopping method which requires 600 GPU hours per task.

6. We conduct the DPS on each task using only 20 architectures. We do not have any prior knowledge of the tasks besides the 20 architecture's ground truth performance since DPS only utilizes a batch of CIFAR-10 images as input. The results are shown in Table 4. Note that though Eproxy underperforms regarding the ranking correlation, it achieves better top-10% retrieve rate compared to other methods. It also tells that the global ranking correlation is not the golden metric for evaluating the performance of proxies since it merely reflects the difference of top architectures. With the help of DPS, the average ranking correlation and top 10% retrieve rate are significantly improved and substantially better than other methods. Compared to the early stopping method, DPS requires 7.6X less regarding GPU hours (>99% time for obtaining the performance of 20 architectures while the DPS only takes 0.5 GPU hour).

	Cls-A	Cls-B	Cls-C	Cls-10c	Seg	Seg-4x	3dDet	Video	Video-p	Avg.
Synflow	0.25 11%	0.05 14%	0.37 20%	0.21 15%	0.43 17%	0.22 9%	0.22 8%	0.45 18%	0.52 17%	0.30 14.3%
NASWOT	0.37 18%	-0.20 4%	-0.15 2%	-0.39 0%	0.50 10%	0.38 8%	0.48 10%	-0.36 1%	-0.36 0%	0.03 6%
ZenNAS	0.41 4%	0.50 0%	0.30 1%	0.25 0%	0.49 6%	0.22 2%	0.26 10%	0.41 0%	0.39 0%	0.36 3%
ZiCo	0.40 4%	0.52 0%	0.31 1%	0.27 0%	0.48 5%	0.21 1%	0.25 9%	0.42 0%	0.40 0%	0.36 2%
Eproxy	0.52 18%	0.06 10%	0.02 10%	0.29 15%	0.38 17%	0.31 13%	0.34 23%	0.31 11%	0.23 11%	0.27 14%
Eproxy + DPS	**0.57** 16%	**0.53** **35%**	0.30 18%	0.48 32%	**0.60** **24%**	**0.51** **13%**	0.39 **29%**	**0.65** **33%**	**0.59** **27%**	**0.51** **25%**
Cls-C Full training (∼4000GPU hrs)	0.29 **24%**	0.51 26%	**1.0** **100%**	**0.53** **34%**	0.21 16%	0.35 26%	0.17 14%	0.35 22%	0.37 25%	n/a N/A

Table 5. Comparison with efficient proxies, and Cls-C full training which requires 4000 GPU hours on NAS-Bench-MR. In 9 tasks, Eproxy achieves superior top-10% retrieval rates compared to recent zero-cost methods like ZenNAS and ZiCo. Eproxy + DPS demonstrates outstanding performance, even when compared to full training on Cls-C.

	RS	NAO	RE	Semi	WeakNAS			Eproxy+DPS		
Queries	2000	2000	2000	1000	200	150	100	150	60	0
Test Acc.	93.64	93.90	93.96	94.01	94.18	94.10	93.69	**94.23**	**93.92**	**93.07**

Table 6. Comparison with predictor-based methods and efficient proxies on NAS-Bench-101. Eproxy+DPS as the fitness function for Regularization Evolutionary Algorithm can find near-optimal architectures with lower queries.

	Random Search	Regularized Evolution	MCTS	LaNAS	WeakNAS	Eproxy+DPS
C10	7782.1	563.2	528.3	247.1	182.1	**58.0 + 20**
C100	7621.2	438.2	405.4	187.5	78.4	**13.7$_T$**
TinyImg	7726.1	715.1	578.2	292.4	268.4	**74.0$_T$**

Table 7. Comparison with predictor-based methods on NAS-Bench-201 regarding the average queries required for retrieving the global optimal architectures. Eproxy+DPS uses substantially lower queries to find the global optimal architectures.

3.4. NAS-Bench-MR

We applied Eproxy and DPS to a complex search space, NAS-Bench-MR [13], with 9 high-resolution tasks such as 3D detection, ImageNet-level classification, segmentation, and video recognition [9, 10, 18, 28]. Approximately 2,500 architectures were benchmarked. Each architecture underwent full training (over 100 epochs) and followed a multi-resolution paradigm, with each network consisting of four stages. Each stage comprised modularized blocks (parallel and fusion modules). Our work is the first to investigate this benchmark with efficient proxies. The results are displayed in Table 5. The full training consumption on the Cls-C task was calculated based on the source code [14], which took approximately 4000 GPU hours. Note that NASWOT, which performs well on NAS-Bench-Trans-Micro, delivers poor performance on most tasks, implying the inconsistent performance of current efficient proxies. Also, we observed that classification rankings are inconsistent with other tasks, such as segmentation and 3D detection. Our Eproxy+DPS experiments suggest that with a 20-architecture set, the ranking correlation and top-10% retrieve rate are considerably improved (**+89%/+78%**).

3.5. End-to-end NAS with Eproxy

We evaluate Eproxy and DPS on the end-to-end NAS tasks to find efficient architectures in the search space.

On **NAS-Bench-101**, we utilize the Eproxy as the fitness function for Regularized Evolutionary (RE) algorithm. Our results are shown in Table 6 compared with NAO [40], Semi-NAS [39], WeakNAS [57]. Note that Eproxy, without any query (near-zero-cost) from the benchmark, can find architectures that are significantly better than current SoTA efficient proxies, Synflow (+ 0.87%) and NASWOT (+3.01%). With 20 architectures for DPS and 40

Method	Test Err. (%)		Params	FLOPS	Search Cost	Searched	Searched
	top-1	top-5	(M)	(M)	(GPU days)	Method	dataset
NASNet-A [66]	26.0	8.4	5.3	564	2000	RL	CIFAR-10
AmoebaNet-C [45]	24.3	7.6	6.4	570	3150	evolution	CIFAR-10
PNAS [36]	25.8	8.1	5.1	588	225	SMBO	CIFAR-10
DARTS(2nd order) [37]	26.7	8.7	4.7	574	4.0	gradient-based	CIFAR-10
SNAS [59]	27.3	9.2	4.3	522	1.5	gradient-based	CIFAR-10
GDAS [15]	26.0	8.5	5.3	581	0.21	gradient-based	CIFAR-10
P-DARTS [8]	24.4	7.4	4.9	557	0.3	gradient-based	CIFAR-10
P-DARTS	24.7	7.5	5.1	577	0.3	gradient-based	CIFAR-100
PC-DARTS [60]	25.1	7.8	5.3	586	0.1	gradient-based	CIFAR-10
TE-NAS [7]	26.2	8.3	6.3	-	0.05	training-free	CIFAR-10
PC-DARTS	24.2	7.3	5.3	597	3.8	gradient-based	ImageNet
ProxylessNAS [5]	24.9	7.5	7.1	465	8.3	gradient-based	ImageNet
TE-NAS [7]	24.5	7.5	5.4	599	0.17	training-free	ImageNet
Eproxy	25.7	8.1	4.9	542	0.02	evolution+proxy	CIFAR-10
Eproxy+DPS$_{\text{T}}$	24.4	7.3	5.3	578	0.06	evolution+proxy	CIFAR-10

Table 8. Comparison with state-of-the-art NAS methods on ImageNet. $_{\text{T}}$ stands for DPS is conducted in NDS search space and directly transferred to the target. Note Eproxy+DPS achieves the best results among NAS methods on CIFAR-10.

queries (total of 60) to retrieve the top architectures during RE, Eproxy+DPS achieves better results than existing SoTA predictor-based NAS WeakNAS with 100 queries (+0.23%). Furthermore, we explore the 70 neighbors of the top architectures (a total of 150 queries) and find architectures with an average of 94.23% accuracy. Note that Semi-NAS with 1000 queries can only reach 94.01%. On **NAS-Bench-201**, we perform the DPS on the CIFAR-10 dataset, and the found proxy is directly transferred to CIFAR-100 and Tiny-ImageNet. We compare with MCTS [52], LaNAS [51], WeakNAS [57]. In Table 7, we show that Eproxy+DPS can find optimal global architectures within the RE search history. Compared to RE, which directly queries the benchmark, our approach reduced 7x/32x/9x query times on three datasets. Compared to predictor-based NAS, Eproxy+DPS also requires fewer queries to discover the optimal architectures. Our results offer an exciting yet promising direction besides pure predictor-based NAS.

DARTS-ImageNet search space We conduct end-to-end search on ImageNet-1k dataset within the DARTS search space as defined in [37]. The network depth is 14 blocks. The input channels are 48, and the FLOPS range from 500M to 600M for the searched architectures. We utilize the 20 samples from the NDS-DARTS search space (not the same search space as the target) and conduct DPS on CIFAR-10 for 200 epochs in one GPU hour. Then we perform the NAS by adopting regularized evolutionary algorithm with the loss of the zero-cost proxy as the fitness function in 0.4 GPU hour. We compare our method with (a) existing works on the DARTS search space [7,8,15,37,59,60] and (b) works on the similar search spaces [5, 36, 45, 66]. The results are shown in Table 8. Eproxy achieves a top-1/5 test error of 25.2%/8.1% using Eproxy with only 0.5 GPU hours for NAS. With DPS, Eproxy explores the architecture with 24.4%/7.3% as a top-1/top5 test error. Eproxy+DPS significantly outperforms existing NAS on CIFAR-10, such as PC-DARTS [60], and achieves a comparable result with NAS on ImageNet, demonstrating Eproxy and DPS's efficiency. By utilizing the existing proxy on another search space, DPS shows the transferability between search spaces.

4. Conclusion

We proposed Eproxy that utilizes a self-supervised few-shot regression task within near-zero cost. The Eproxy is benefited from the barrier layer that significantly improves the complexity of the proxy task. To overcome the drawbacks of current efficient proxies that are not adaptive to various tasks/search spaces, we proposed DPS incorporating various settings and hyperparameters in a proxy search space and leveraging REA to conduct efficient exploration. Our experiments on numerous NAS benchmarks demonstrate that Eproxy is a robust, efficient proxy. Moreover, with the help of DPS, Eproxy achieves state-of-the-art results and outperforms existing state-of-the-art efficient proxies, early stopping methods and predictor-based NAS. Our work significantly ameliorates the inconsistency of efficient proxies and sets up a series of solid baselines while pointing out a novel direction for the NAS community.

5. Acknowledgement

This work is supported in part by the National Science Foundation awards #2229873 and #2235364. We thank all reviewers for valuable discussions and feedback.

References

[1] Mohamed S Abdelfattah, Abhinav Mehrotra, Łukasz Dudziak, and Nicholas D Lane. Zero-cost proxies for lightweight nas. *arXiv preprint arXiv:2101.08134*, 2021. 1, 6

[2] Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016. 4

[3] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *European conference on computer vision*, pages 850–865. Springer, 2016. 4

[4] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019. 1

[5] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018. 9

[6] Hanlin Chen, Ming Lin, Xiuyu Sun, and Hao Li. Nas-bench-zero: A large scale dataset for understanding zero-shot neural architecture search. 2021. 2

[7] Wuyang Chen, Xinyu Gong, and Zhangyang Wang. Neural architecture search on imagenet in four gpu hours: A theoretically inspired perspective. *arXiv preprint arXiv:2102.11535*, 2021. 9

[8] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1294–1303, 2019. 9

[9] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016. 8

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 6, 8

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 1

[12] Debadeepta Dey, Shital Shah, and Sebastien Bubeck. Ranking architectures by feature extraction capabilities. In *8th ICML Workshop on Automated Machine Learning (AutoML)*, 2021. 5

[13] Mingyu Ding, Yuqi Huo, Haoyu Lu, Linjie Yang, Zhe Wang, Zhiwu Lu, Jingdong Wang, and Ping Luo. Learning versatile neural architectures by propagating network codes. *arXiv preprint arXiv:2103.13253*, 2021. 2, 5, 8

[14] Mingyu Ding, Yuqi Huo, Haoyu Lu, Linjie Yang, Zhe Wang, Zhiwu Lu, Jingdong Wang, and Ping Luo. Learning Versatile Neural Architectures by Propagating Network Codes. https://github.com/dingmyu/NCP, 2023. 8

[15] Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1761–1770, 2019. 9

[16] Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations (ICLR)*, 2020. 2

[17] Yawen Duan, Xin Chen, Hang Xu, Zewei Chen, Xiaodan Liang, Tong Zhang, and Zhenguo Li. Transnas-bench-101: Improving transferability and generalizability of cross-task neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5251–5260, 2021. 5

[18] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 8

[19] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 4

[20] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010. 4

[21] Xinyu Gong, Shiyu Chang, Yifan Jiang, and Zhangyang Wang. Autogan: Neural architecture search for generative adversarial networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3224–3234, 2019. 1

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. 4

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1

[24] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 1

[25] Mojan Javaheripi, Shital Shah, Subhabrata Mukherjee, Tomasz L Religa, Caio CT Mendes, Gustavo H de Rosa, Sebastien Bubeck, Farinaz Koushanfar, and Debadeepta Dey. Litetransformersearch: Training-free on-device search for efficient autoregressive language models. *arXiv preprint arXiv:2203.02094*, 2022. 4

[26] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 2

[27] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012. 1

[28] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *2011 Inter-*

national conference on computer vision, pages 2556–2563. IEEE, 2011. 8

[29] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8971–8980, 2018. 4

[30] Changlin Li, Tao Tang, Guangrun Wang, Jiefeng Peng, Bing Wang, Xiaodan Liang, and Xiaojun Chang. Bossnas: Exploring hybrid cnn-transformers with block-wisely self-supervised neural architecture search. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12281–12291, 2021. 1

[31] Guihong Li, Yuedong Yang, Kartikeya Bhardwaj, and Radu Marculescu. Zico: Zero-shot nas via inverse coefficient of variation on gradients. *arXiv preprint arXiv:2301.11300*, 2023. 1, 6

[32] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31, 2018. 5

[33] Yuhong Li, Cong Hao, Pan Li, Jinjun Xiong, and Deming Chen. Generic neural architecture search via regression. *Advances in Neural Information Processing Systems*, 34, 2021. 2, 3, 4

[34] Ming Lin, Pichao Wang, Zhenhong Sun, Hesen Chen, Xiuyu Sun, Qi Qian, Hao Li, and Rong Jin. Zen-nas: A zero-shot nas for high-performance deep image recognition. *arXiv preprint arXiv:2102.01063*, 2021. 1, 6

[35] Chenxi Liu, Piotr Dollár, Kaiming He, Ross Girshick, Alan Yuille, and Saining Xie. Are labels necessary for neural architecture search? In *European Conference on Computer Vision*, pages 798–813. Springer, 2020. 1

[36] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *Proceedings of the European conference on computer vision (ECCV)*, pages 19–34, 2018. 9

[37] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018. 1, 9

[38] Zhijian Liu, Haotian Tang, Shengyu Zhao, Kevin Shao, and Song Han. Pvnas: 3d neural architecture search with point-voxel convolution. *arXiv preprint arXiv:2204.11797*, 2022. 1

[39] Renqian Luo, Xu Tan, Rui Wang, Tao Qin, Enhong Chen, and Tie-Yan Liu. Semi-supervised neural architecture search. *Advances in Neural Information Processing Systems*, 33:10547–10557, 2020. 8

[40] Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu. Neural architecture optimization. *Advances in neural information processing systems*, 31, 2018. 8

[41] Joe Mellor, Jack Turner, Amos Storkey, and Elliot J Crowley. Neural architecture search without training. In *International Conference on Machine Learning*, pages 7588–7598. PMLR, 2021. 1, 6

[42] Xuefei Ning, Changcheng Tang, Wenshuo Li, Zixuan Zhou, Shuang Liang, Huazhong Yang, and Yu Wang. Evaluating efficient performance estimators of neural architectures. *Advances in Neural Information Processing Systems*, 34, 2021. 2, 4

[43] Jiefeng Peng, Jiqi Zhang, Changlin Li, Guangrun Wang, Xiaodan Liang, and Liang Lin. Pi-nas: Improving neural architecture search by reducing supernet training consistency shift. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12354–12364, 2021. 1

[44] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10428–10436, 2020. 5, 6

[45] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789, 2019. 1, 5, 9

[46] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 1

[47] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997. 1

[48] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1

[49] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019. 1

[50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 1

[51] Linnan Wang, Saining Xie, Teng Li, Rodrigo Fonseca, and Yuandong Tian. Sample-efficient neural architecture search by learning actions for monte carlo tree search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 9

[52] Linnan Wang, Yiyang Zhao, Yuu Jinnai, Yuandong Tian, and Rodrigo Fonseca. Alphax: exploring neural architectures with deep neural networks and monte carlo tree search. *arXiv preprint arXiv:1903.11059*, 2019. 9

[53] Ning Wang, Yang Gao, Hao Chen, Peng Wang, Zhi Tian, Chunhua Shen, and Yanning Zhang. Nas-fcos: Fast neural architecture search for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11943–11951, 2020. 1

[54] Yu Weng, Tianbao Zhou, Yujie Li, and Xiaoyu Qiu. Nas-unet: Neural architecture search for medical image segmentation. *IEEE Access*, 7:44247–44257, 2019. 1

[55] Colin White, Mikhail Khodak, Renbo Tu, Shital Shah, Sébastien Bubeck, and Debadeepta 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/. 4, 6

[56] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10734–10742, 2019. 1, 4

[57] Junru Wu, Xiyang Dai, Dongdong Chen, Yinpeng Chen, Mengchen Liu, Ye Yu, Zhangyang Wang, Zicheng Liu, Mei Chen, and Lu Yuan. Stronger nas with weaker predictors. *Advances in Neural Information Processing Systems*, 34:28904–28918, 2021. 8, 9

[58] Zhanghao Wu, Zhijian Liu, Ji Lin, Yujun Lin, and Song Han. Lite transformer with long-short range attention. *arXiv preprint arXiv:2004.11886*, 2020. 1

[59] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926*, 2018. 9

[60] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. Pc-darts: Partial channel connections for memory-efficient architecture search. *arXiv preprint arXiv:1907.05737*, 2019. 9

[61] Zhi-Qin John Xu, Yaoyu Zhang, Tao Luo, Yanyang Xiao, and Zheng Ma. Frequency principle: Fourier analysis sheds light on deep neural networks. *arXiv preprint arXiv:1901.06523*, 2019. 4

[62] Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. Nas-bench-101: Towards reproducible neural architecture search. In *International Conference on Machine Learning*, pages 7105–7114. PMLR, 2019. 2, 5

[63] Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3712–3722, 2018. 6

[64] Miao Zhang, Steven Su, Shirui Pan, Xiaojun Chang, Wei Huang, Bin Yang, and Gholamreza Haffari. Differentiable architecture search meets network pruning at initialization: A more reliable, efficient, and flexible framework. *arXiv preprint arXiv:2106.11542*, 2021. 1

[65] Xuanyang Zhang, Pengfei Hou, Xiangyu Zhang, and Jian Sun. Neural architecture search with random labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10907–10916, 2021. 1

[66] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018. 1, 9