EFFICIENT PROXY FOR NAS IS EXTENSIBLE NOW

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural Architecture Search (NAS) has become a de facto approach in the recent trend of AutoML to design deep neural networks (DNNs). Efficient or near-zero-cost NAS proxies are further proposed to address the demanding computational issues of NAS, where each candidate architecture network only requires one iteration of backpropagation. The values obtained from the proxies are considered the predictions of architecture performance on downstream tasks. However, two significant drawbacks hinder the extended usage of Efficient NAS proxies. (1) Efficient proxies are not adaptive to various search spaces. (2) Efficient proxies are not extensible to multi-modality downstream tasks. Based on the observations, we design a Extensible proxy (Eproxy) that utilizes self-supervised, few-shot training (i.e., 10 iterations of backpropagation) which yields near-zero costs. The key component that makes Eproxy efficient is an untrainable convolution layer termed barrier layer that add the non-linearities to the optimization spaces so that the Eproxy can discriminate the performance of architectures in the early stage. Furthermore, to make Eproxy adaptive to different downstream tasks/search spaces, we propose a Discrete Proxy Search (DPS) to find the optimized training settings for Eproxy with only handful of benchmarked architectures on the target tasks. Our extensive experiments confirm the effectiveness of both Eproxy and Eproxy+DPS. On NAS-Bench-101 ($\sim$423k architectures), Eproxy achieves 0.65 as the spearman ρ. In contrast, the previous best zero-cost method achieves 0.45. On NDS-ImageNet search spaces, Eproxy+DPS delivers 0.73 Spearman ρ average ranking correlation while the previous efficient proxy only achieves 0.47. On NAS-Bench-Trans-Micro search space (7 tasks), Eproxy+DPS delivers comparable performance with early stop methods which requires 660 GPU hours per task. For the end-to-end task such as DARTS-ImageNet-1k, our method delivers better results compared to NAS performed on CIFAR-10 while only requiring a GPU hour with a single batch of CIFAR-10 images.

1 INTRODUCTION

As deep neural networks (DNNs) find uses in a wide range of applications, such as computer vision (Krizhevsky et al., 2012; Simonyan & Zisserman, 2014; He et al., 2016; Redmon et al., 2016) and natural language processing (Vaswani et al., 2017; Schuster & Paliwal, 1997; Hochreiter & Schmidhuber, 1997; Wu et al., 2020; Devlin et al., 2018), Neural Architecture Search (NAS) (Zoph et al., 2018; Real et al., 2019; Tan et al., 2019; Cai et al., 2019; Liu et al., 2018b) has become an increasingly important technique to automate the design of neural architectures for different tasks (Weng et al., 2019; Wang et al., 2020b; Liu et al., 2022; Gong et al., 2019). Recent progress in NAS has demonstrated superior results, surpassing those of human designs (Zoph et al., 2018; Wu et al., 2019; Tan et al., 2019). However, one major hurdle for NAS is its high computation cost. For example, the seminal work of NAS (Zoph et al., 2018) 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 the 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 limitation of the search space, the weight-sharing networks to reduce the training cost, and the efficient proxies for evaluating the candidate architectures.

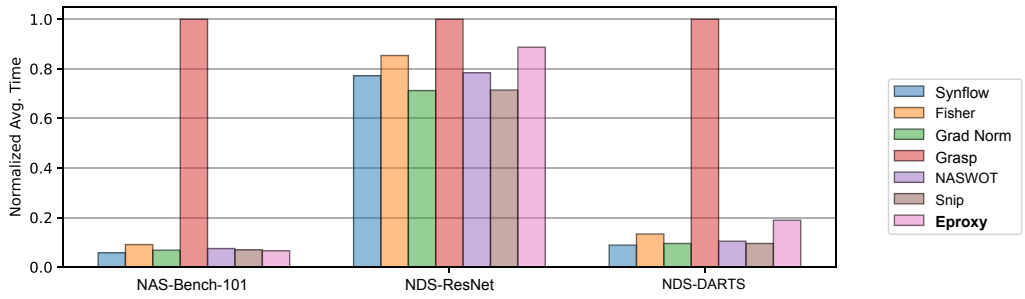


Figure 1: Comparison of Eproxy with six efficient proxies regarding evaluation speed on NAS-Bench-101, NDS-ResNet, and NDS-DARTS search spaces. The normalized average time is plotted.

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. For example, Mellor et al. (2021) delivered NASWOT to analyze the activations of an untrained network as a proxy and demonstrated some promising results. Abdelfattah et al. (2021) proposed various proxies, such as gradients normalization (Grad norm), one-shot pruning based on a saliency metric computed at initialization (Tanaka et al., 2020; Wang et al., 2020a; Lee et al., 2018), and Fisher (Theis et al., 2018) that performs channel pruning by removing activation channels that are estimated to have the most negligible effect on the loss. The above efficient proxies, however, have two significant drawbacks. First, the quality of efficient proxies varies widely for different search spaces. Most of the proxies deliver high correlations with search space limited in the small NAS Benchmarks, while in real-life applications, the size of search spaces are order-of-magnitude larger than the tabular benchmarks'. For example, Synflow achieves high ranking correlation on NAS-Bench-201 Dong & Yang (2020) (0.74 Spearman ρ) but performan poorly on NAS-Bench-101 (Ying et al., 2019) (0.37) which is 27X larger than NAS-Bench-201. (2) Efficient proxies are not extensible to multi-modality downstream tasks. One concern is that they are implicitly designed for CIFAR-10-level classification tasks where proxies deliver promising prediction results. For example, NASWOT fails (0.03 ρ as the average ranking correlation) on NAS-Bench-MR (Ding et al., 2021) (9 real-world tasks). Moreover, most efficient proxies apply specified algorithms such as pruning to transform the weights of architectures into prediction values. The fixed algorithm limits the adaptability of proxies towards tasks beyond classification. Besides, some ZC proxies introduce unknown bias for their preferences for certain neural architectures (Chen et al., 2021a). It has been shown empirically and theoretically (Ning et al., 2021) that Synflow prefers large models.

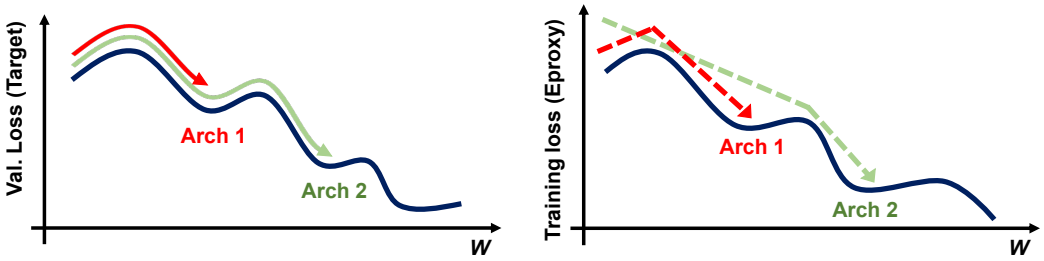
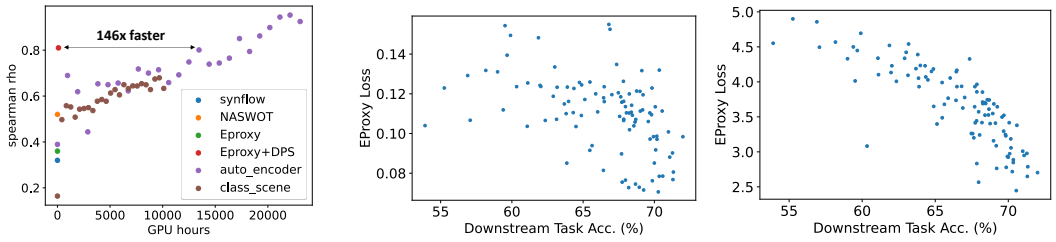


Figure 2: Illustration of the validation losses of two architectures on downstream task(left). A sophisticated few-shot proxy (right) can reflect the actual performance of architectures.

This work introduces a new efficient proxy termed Extensible proxy (Eproxy) from a different angle. Unlike previous efficient proxies, Eproxy utilizes few-shot spatial-level regression on a set of image-label pairs (see Illustration in Fig. 3). The labels are 2D synthetic features since spatial-level regression is more challenging than one-hot classification on a tiny dataset, i.e., a batch of image-label pairs as Li et al. (2021) suggest. The key component of the 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. ($\sim$+0.57 ρ ranking correlation on NAS-Bench-101.) We find that the barrier layer increases the complexity of the optimization space. Hence, poor-performance archi-

tectures 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 that makes Eproxy multi-modalities. We term the search method Discrete Proxy Search (DPS) (The performance of DPS are shown in Fig. 3). Notably, besides the evaluation performance of a handful of architectures, DPS does not need to use any task-specific information (in our experiment, we only use a single batch of CIFAR-10 (Krizhevsky et al., 2009) images throughout all the experiments).



(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.

We summarize our contributions as follows:

- We propose an efficient proxy task with the barrier layer that utilizes a few-shot self-supervised regression. The task adopts only one batch of images in CIFAR-10-level dataset (not necessarily from the target training dataset). It uses the synthetic labels to evaluate architectures. Eproxy significantly speeds up the traditional early stopping evaluation process while maintaining the high ranking correlation.
- We propose the downstream-task/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/downstream-task aware.
- We provide thorough experiments to evaluate the performance of Eproxy and Eproxy boosted by DPS on more than 30 search spaces/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 and benefit the NAS community.

2 Our Approaches

In Sec. 2.1, we introduce the Eproxy for efficient network evaluation; in Sec. 2.2, we discuss how to find a downstream-task/search-space-aware Eproxy via 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 (Li et al., 2021) 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

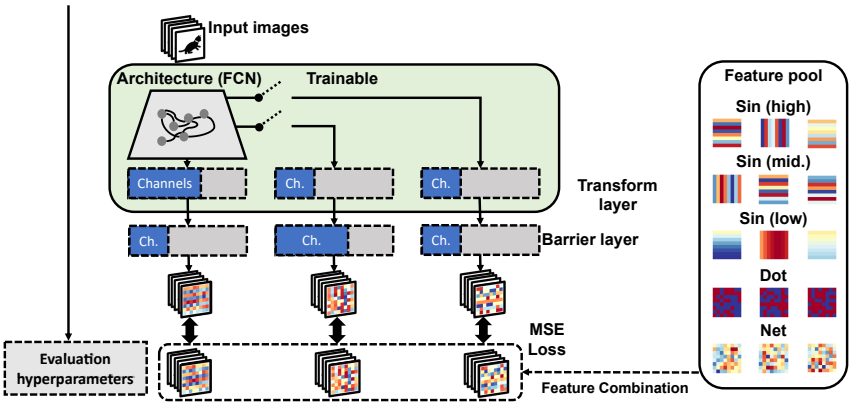


Figure 4: The design of Eproxy and the searchable components. Dotted line: The configurable components for Discrete Proxy Search. Green block: Trainable components. The configurability of Eproxy can be further utilized by DPS to target search spaces/downstream tasks.

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 usually obtained by utilizing the architecture without a task-specified head in the downstream tasks. For example, the classfier network with the classification (average pooling and linear layer) head removed. w_a and w_t are the weights of architecture for evaluation and the weights of the transform layer (a convolution module) 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 FCN (Fig. 4, 'Net'). We interpret that Eproxy conducts a few-shot, tiny knowledge-distillation task from an untrained teacher network.

2.2 Discrete Proxy Search

Since Eproxy provides abundant configurable hyperparameters and utilizes data-agnostic spatial labels, the different settings can be naturally adjusted for tasks/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:

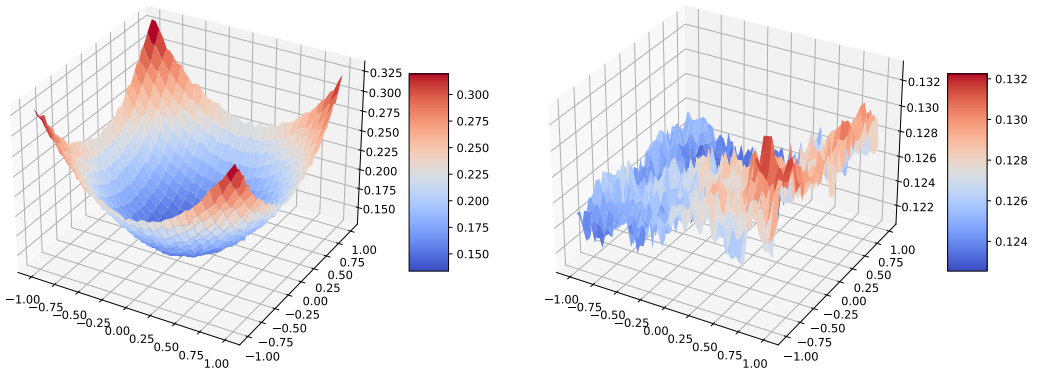
1. Transform and barrier layer: Both layers can have kernel size selected from $\{1, 3, 7\}$, and the channel number c_{mid} can be selected from 16 to 512 geometrically with 2 as a multiplier.
2. Feature combination: a) Untrained FCN outputs. The experiment results show that an untrained network's output features can be powerful for evaluating architectures on numerous tasks/search spaces. 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 (Li et al., 2021; Xu et al., 2019b). 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 (Girshick, 2015), segmentation (Bertinetto et al., 2016), tracking (Bertinetto et al., 2016; Li et al., 2018a). For more details, please refer to the Appendix. 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 (He et al., 2015) and Xavier (Glorot & Bengio, 2010) with either Gaussian or Uniform initialization (total 4 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 (Alain & Bengio, 2016). Thus, monitoring the early stages can give more flexibility for adapting Eproxy to different tasks.
5. FLOPS: As works (Javaheripi et al., 2022; Wu et al., 2019; Ning et al., 2021; White et al., 2022) 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 $\mathcal{L} \cdot (1 + \alpha \cdot \text{norm(FLOPS)})$. 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 5e15. We utilize the regularization evolutionary algorithm (REA) (Real et al., 2019) to conduct the exploration efficiently. 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 considered 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.

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 (Ying et al., 2019), 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 (Dey et al., 2021) to evaluate the proposed method on **NDS** (Radosavovic et al., 2020) (11 search spaces on CIFAR-10, 8 search spaces on ImageNet), **NAS-Bench-Trans-Micro** Duan et al. (2021) (7 tasks), and **NAS-Bench-MR** Ding et al. (2021) (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.



(a) Regression without the barrier.

(b) Regression with the barrier (Eproxy).

Figure 5: The loss surfaces of regression task with/without the barrier.

3.1 ABLATION STUDY ON NAS-BENCH-101

We study the effectiveness of our barrier layer in this section. We use the tool from (Li et al., 2018b) to visualize the loss surface of an architecture selected randomly from NAS-Bench-101 on

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.

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

Table 2: Comparison with efficient proxies on NAS-Bench-101 using the Spearman ρ and top-10% retrieve rate.

our few-shot regression task. Figure 5 (a) 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 architecture (See A.7 for more visualization). 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.

Next, we sample 20 architectures from NAS-Bench-101 and evaluate DPS. We conduct DPS for 200 epochs, and the total run time is $\sim$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. Eproxy significantly outperforms existing zero-cost proxies by a large margin. For example, Synflow, considered the stable proxy, achieves 0.45, NASWOT only achieves 0.38, Eproxy achieves 0.65 (without DPS), and Eproxy + DPS achieves 0.69. Regarding the top-10% retrieve rate, Eproxy + DPS retrieves more architectures than DPS (38% vs. 31%). The results support the efficiency and effectiveness of DPS. Meanwhile, Fig. 1 confirms that using Eproxy can achieve the same evaluation speed compared with other efficient proxies.

3.2 NDS

Mellor et al. (2021) utilizes an interesting and practical dataset named Network Design Spaces (NDS), where the original paper aims to compare the search spaces themselves. The NDS is perfect for evaluating efficient proxies in more complex search spaces. For example, researchers benchmark 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** Deng et al. (2009). We show the results in Table 3. Compared to NASWOT (Mellor et al., 2021), Eproxy (without DPS) achieves on-a-par results 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, respec-

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%
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%
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 is conducted on CIFAR-10 and directly transferred to **ImageNet**.

	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%
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$_{\sim \text{ 660GPU hrs/task}}$	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. Eproxy+DPS outperforms efficient proxies and early stopping method.

tively. The ImageNet experiment demonstrates the efficiency by utilizing the architectures trained on down-scaled dataset (CIFAR-10) for DPS.

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** Zamir et al. (2018) 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 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. We compare our method with NASWOT, Synflow, and the early stopping method shown in Table 4. Note that though Eproxy underperforms regarding the ranking correlation, it achieves an 89% higher top-10% retrieve rate compared to Synflow. 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).

3.4 NAS-BENCH-MR

We try the Eproxy and DPS on a more complex search space, NAS-Bench-MR (Ding et al., 2021), with **9 high-resolution tasks** such as 3d detection, ImageNet-level classification, segmentation, and video recognition Deng et al. (2009); Cordts et al. (2016); Geiger et al. (2012); Kuehne et al. (2011).

	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%
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 on NAS-Bench-MR. Eproxy+DPS is comparable with the full training on Cls-C task.

	RS	NAO	RE	Semi	WeakNAS			Synflow	NASWOT	Eproxy+DPS		
Queries	2000	2000	2000	1000	200	150	100	0	0	150	60	0
Test Acc.	93.64	93.90	93.96	94.01	94.18	94.10	93.69	92.20	90.06	**94.23**	**93.92**	**93.07**

Table 6: Comparison with predictor-based methods and efficient proxies on NAS-Bench-101. Eproxy+DPS can find near-optimal architectures with lower queries.

Randomly sampled ∼2,500 architectures are evaluated on the tasks from the entire search space. Each architecture is fully trained (>100 epochs) and follows a multi-resolution paradigm, where each network contains four stages. Each stage comprises modularized blocks (parallel and fusion modules). Hence, the benchmark is unprecedentedly complicated. Our work is the first to investigate this benchmark with efficient proxies. We compared Eproxy and Eproxy+DPS with NASWOT, Synflow, and full training on Cls-C task (4000GPU hrs [1]). The results are shown in Table 5. 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, aiming to find high-performance architectures within the search space.

	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	$\mathbf{13.7_T}$
TinyImg	7726.1	715.1	578.2	292.4	268.4	$\mathbf{74.0_T}$

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.

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 (Luo et al., 2018), Semi-NAS (Luo et al., 2020), WeakNAS (Wu et al., 2021), Synflow (Abdelfattah et al., 2021), NASWOT (Mellor et al., 2021). 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 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.

[1] https://github.com/dingmyu/NCP

Method	Test Err. (%)		Params	FLOPS	Search Cost	Searched	Searched
	top-1	top-5	(M)	(M)	(GPU days)	Method	dataset
NASNet-A Zoph et al. (2018)	26.0	8.4	5.3	564	2000	RL	CIFAR-10
AmoebaNet-C Real et al. (2019)	24.3	7.6	6.4	570	3150	evolution	CIFAR-10
PNAS Liu et al. (2018a)	25.8	8.1	5.1	588	225	SMBO	CIFAR-10
DARTS(2nd order) Liu et al. (2018b)	26.7	8.7	4.7	574	4.0	gradient-based	CIFAR-10
SNAS Xie et al. (2018)	27.3	9.2	4.3	522	1.5	gradient-based	CIFAR-10
GDAS Dong & Yang (2019)	26.0	8.5	5.3	581	0.21	gradient-based	CIFAR-10
P-DARTS Chen et al. (2019)	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 Xu et al. (2019a)	25.1	7.8	5.3	586	0.1	gradient-based	CIFAR-10
TE-NAS Chen et al. (2021b)	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 Cai et al. (2018)	24.9	7.5	7.1	465	8.3	gradient-based	ImageNet
TE-NAS Chen et al. (2021b)	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$_{\mathrm{T}}$	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. $_{\mathrm{T}}$ 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.

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 (Wang et al., 2019), LaNAS (Wang et al., 2021), WeakNAS (Wu et al., 2021). 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.

Open DARTS-ImageNet search space On DARTS search space (Liu et al., 2018b), we perform the end-to-end search on ImageNet-1k (Deng et al., 2009) dataset. The networks' depth (number of micro-searching blocks) is 14. The input channel number is 48, and architectures are with FLOPs between 500M to 600M. 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 a 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 Liu et al. (2018b); Xie et al. (2018); Dong & Yang (2019); Chen et al. (2019); Xu et al. (2019a); Chen et al. (2021b) and (b) works on the similar search spaces Zoph et al. (2018); Real et al. (2019); Liu et al. (2018a); Cai et al. (2018). 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, and achieves a comparable result with NAS on ImageNet, demonstrating Eproxy and DPS's efficiency. By utilizing the existing performance of architectures on another dataset/search space, DPS shows the transferability between tasks and search spaces.

4 Conclusion

In this work, 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.

REFERENCES

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

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

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*, pp. 850–865. Springer, 2016.

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

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.

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

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*, 2021b.

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*, pp. 1294–1303, 2019.

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*, pp. 3213–3223, 2016.

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*, pp. 248–255. Ieee, 2009.

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.

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

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.

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*, pp. 1761–1770, 2019.

Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations (ICLR)*, 2020. URL `https://openreview.net/forum?id=HJxyZkBKDr`.

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*, pp. 5251–5260, 2021.

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.

Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

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*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

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*, pp. 3224–3234, 2019.

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*, pp. 1026–1034, 2015.

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*, pp. 770–778, 2016.

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

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.

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

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

Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *2011 International conference on computer vision*, pp. 2556–2563. IEEE, 2011.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018.

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*, pp. 8971–8980, 2018a.

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, 2018b.

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.

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)*, pp. 19–34, 2018a.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018b.

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.

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

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.

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

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.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *International conference on machine learning*, pp. 4095–4104. PMLR, 2018.

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*, pp. 10428–10436, 2020.

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, pp. 4780–4789, 2019.

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*, pp. 779–788, 2016.

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

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

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*, pp. 2820–2828, 2019.

Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. *Advances in Neural Information Processing Systems*, 33:6377–6389, 2020.

Lucas Theis, Iryna Korshunova, Alykhan Tejani, and Ferenc Huszár. Faster gaze prediction with dense networks and fisher pruning. *arXiv preprint arXiv:1801.05787*, 2018.

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.

Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. *arXiv preprint arXiv:2002.07376*, 2020a.

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.

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.

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*, pp. 11943–11951, 2020b.

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

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. URL `https://iclr-blog-track.github.io/2022/03/25/zero-cost-proxies/`. https://iclr-blog-track.github.io/2022/03/25/zero-cost-proxies/.

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*, pp. 10734–10742, 2019.

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.

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

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500, 2017.

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

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*, 2019a.

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*, 2019b.

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*, pp. 7105–7114. PMLR, 2019.

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*, pp. 3712–3722, 2018.

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*, pp. 8697–8710, 2018.

A Appendix

A.1 Experiment Setup

Eproxy The learning rate is 1.0, and the weight decay is $1\mathrm{e}{-5}$. Each architecture is trained for ten iterations with 16 images randomly sampled from the CIFAR-10 training set as a mini-batch (tiny dataset). The SGD optimizer is used for training.

DPS The total evolution cycle is 200. The number of architectures sampled for ranking is 20. The population size is 40. The sample size is 10. The mutation rate is 0.2.

Search space	NB101	NB201	DARTS	DARTS-fix-w-d	Amoeba
Avg. Eval. Time (ms)	414.1	324.0	719.2	1198.3	1191.3
GPU Util. (MB)	4137	1603	3221	2275	3365
Search space	ENAS	ENAS-fix-w-d	NASNet	PNAS	PNAS-fix-w-d
Avg. Eval. Time (ms)	908.2	1408.2	878.7	1041.4	1824.7
GPU Util. (MB)	3245	2577	3129	3391	3447
Search space	ResNet	ResNeXt-A	ResNeXt-B	NAS-Bench-Trans-Micro	NAS-Bench-MR
Avg. Eval. Time (ms)	242.3	314.5	298.7	355.2	1011.9
GPU Util. (MB)	2765	2423	2777	2081	4229

Table 9: Average time for evaluating an architecture with Eproxy in the target search space and Maximum GPU utilization. The results suggest that Eproxy is efficient and computation-friendly.

A.2 GPU BENCHMARK

We benchmark the average evaluation time for architecture with Eproxy and GPU utilization on different search spaces (shown in Table 9). For DPS, it's straightforward to estimate the total time. For example, if we conduct DPS on NDS-DARTS search space with 20 architectures to get each proxy's ranking correlation and 200 total evolution cycles, the time is $\sim 20 \times 200 \times 0.72 = 2880$ seconds. All experiments are done on a single A6000 GPU.

A.3 SEARCH SPACES

NAS-Bench-101 (Ying et al., 2019): 423K CNN architectures are trained on CIFAR-10 dataset.

NAS-Bench-201 (Dong & Yang, 2020): 15625 CNN architectures are trained on CIFAR-10/CIFAR-100/TinyImageNet.

NDS dataset (Radosavovic et al., 2020): **DARTS**: A DARTS (Liu et al., 2018b) style search space including 5000 sampled architectures trained on CIFAR-10. **DARTS-fix_w_d**: A DARTS style search space with fixed width and depth including 5000 sampled architectures trained on CIFAR-10. **AmoebaNet**: An AmoebaNet (Real et al., 2019) style search space including 4983 sampled architectures trained on CIFAR-10. **ENAS**: An ENAS (Pham et al., 2018) style search space including 4999 sampled architectures trained on CIFAR-10. **ENAS-fix_w_d**: An ENAS style search space with fixed width and depth including 5000 sampled architectures trained on CIFAR-10. **NASNet**: A NASNet (Zoph et al., 2018) style search space including 4846 sampled architectures trained on CIFAR-10. **PNAS**: A PNAS (Liu et al., 2018a) style search space including 4999 sampled architectures trained on CIFAR-10. **PNAS-fix_w_d**: A PNAS style search space with fixed width and depth including 4559 sampled architectures trained on CIFAR-10. **ResNet**: A ResNet (He et al., 2016) style search space including 25000 sampled architectures trained on CIFAR-10. **ResNeXt-A**: A ResNeXt Xie et al. (2017) style search space including 24999 sampled architectures trained on CIFAR-10. **ResNeXt-B**: Another ResNeXt style search space including 25508 sampled architectures trained on CIFAR-10. **DARTS_in**: A DARTS style search space including 121 sampled architectures trained on ImageNet-1k. **DARTS-fix_w_d-in**: A DARTS style search space with fixed width and depth including 499 sampled architectures trained on ImageNet-1k. **Amoeba_in**: An AmoebaNet style search space including 124 sampled architectures trained on ImageNet-1k. **ENAS_in**: A ENAS style search space including 117 sampled architectures trained on ImageNet-1k. **NASNet_in**: A NASNet style search space including 122 sampled architectures trained on ImageNet-1k. **PNAS_in**: A PNAS style search space including 119 sampled architectures trained on ImageNet-1k. **ResNeXt-A_in**: A ResNeXt style search space including 130 sampled architectures trained on ImageNet-1k. **ResNeXt-B_in**: Another ResNeXt style search space including sampled 164 architectures trained on ImageNet-1k.

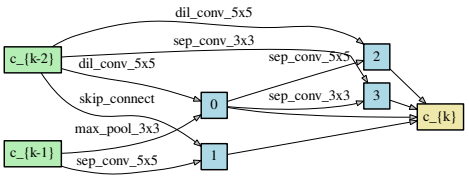
NAS-Bench-Trans-Micro Duan et al. (2021): A NAS-Bench-201 style search space including 4096 architectures trained on 7 different tasks on the subsets of Taskonomy dataset (Zamir et al., 2018). Tasks including: **Object Classification** for 75 classes of objects. **Scene Classification** for 47 classes of scenes. **Room Layout** for estimating and aligning a 3D bounding box by utilizing a 9-dimension vector. **Jigsaw Content Prediction** by dividing the input image into 9 patches and shuffling accord-

ing to one of 1000 preset permutations. **Semantic Segmentation** for 17 semantic classes. **Autoencoding** for reconstructing the input images.

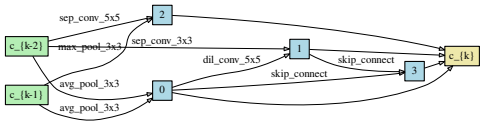
NAS-Bench-MR (Ding et al., 2021): A complex search space for multi-resolution networks including 2507 trained architectures on 9 different tasks. Tasks including: **ImageNet-50-1000 (Cls-A)** with 50 classes and 1000 samples from each class from ImageNet-1k. **ImageNet-50-100 (Cls-B)** with 50 classes and 100 samples from each class from ImageNet-1k. **ImageNet-10-1000 (Cls-A)** with 10 classes and 1000 samples from each class from ImageNet-1k. **ImageNet-10c** same as Cls-A but architectures are trained for 10 epochs. **Seg** for Cityscapes dataset (Cordts et al., 2016). **Seg-4x** for Cityscapes dataset with 4x downsampled resolution. **3dDet** on KITTI dataset (Geiger et al., 2012). **Video** for HMDB51 dataset Kuehne et al. (2011). **Video-p** for HMDB51 but architectures are pretrained with ImageNet-50-1000.

A.4 Searched Architectures

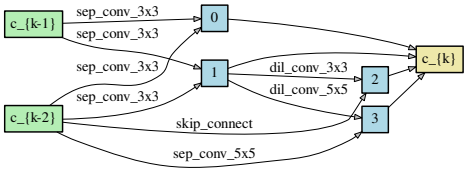
The searched architectures for DARTS-ImageNet search space are shown in Fig 6.



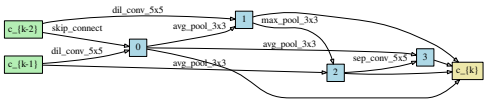
(a) Eproxy Normal Cell



(b) Eproxy Reduction Cell



(c) Eproxy+DPS Normal Cell



(d) Eproxy+DPS Reduction Cell

Figure 6: Visualize the architecture found by Eproxy and Eproxy+DPS on ImageNet-DARTS search space.

A.5 Pseudo Code for Eproxy

```
def Eproxy(model, barrier, img, label, t_iter = 10):
    # img shape: B, C = 3, W_in, H_in
    # label shape: B,  C_out, W_out, H_out
    optimizer = torch.optim.SGD(model.parameters(),
                                lr=1.0,
                                momentum=0.9,
                                weight_decay=4e-5)
    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 - label)**2).mean()
        optimizer.zero_grad()
        loss_m.backward()
        optimizer.step()
    return loss
```

Listing 1: Pseudo PyTorch-sytle Paszke et al. (2019) code for Eproxy.

A.6 Pseudo Code for DPS

```
def DPS(archs_accs, cycle, population = 40, sample = 10, mutation_rate = 0.2):
    # len(archs_accs): 20
    # config: including lr, channel number, feature combination, etc.
    config_history = []
    rea = REAEngine(population, sample, mutation_rate)
    # generate initial pool
    for _ in range(population):
        config = rea.get_random_config()
        rank = rea.get_rank(config, archs_accs)
        config_history.append({'config': config, 'rank': rank})
    # evolution
    for _ in range(cycle):
        new_config = rea.get_mutate_config()
        rank = rea.get_rank(new_config, archs_accs)
        config_history.append({'config': new_config, 'rank': rank})
        # rea.get_config_pool().size(): 40
        # rea.get_config_pool_rank().max(): the proxy in the pool with highest ranking correlation on the archs_accs set)
    return config_history
```

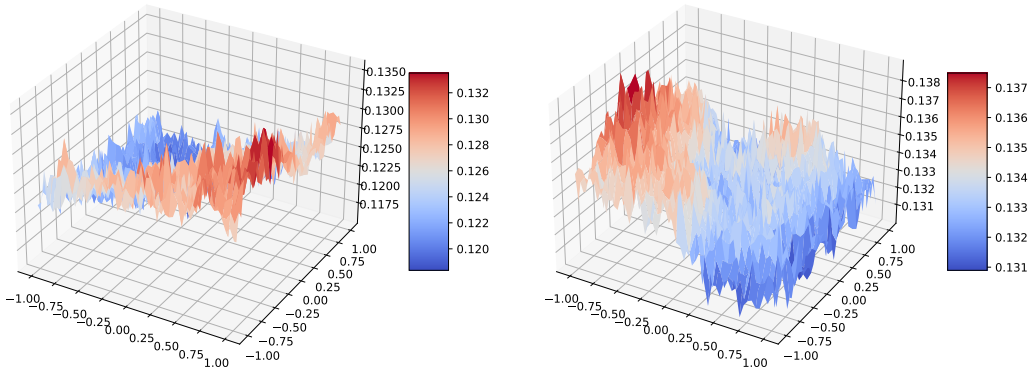
Listing 2: Pseudo PyTorch-sytle code for DPS.

A.7 More Loss landscapes

We listed more loss landscapes from the best and the worst models in NAS-Bench-101 (Ying et al., 2019) search space on our proxy task, either with or without the barrier in Fig. 7 and Fig. 8. From Fig. 8, we can observe that the best model has a much smoother loss surface than the worst model. From Fig. 7, we can observe that the best model's can achieve lower loss compared to worst model even though the loss surface is sophisticated. Besides, the loss surfaces are significantly different which means the optimization directions for both models are distinctive. We can also observe from Fig. 8 that the best and worst models have similar convexity and shape, which makes the proxy task produce a much worse ranking correlation score compared with the proxy task that uses the barrier.

A.8 Limitations

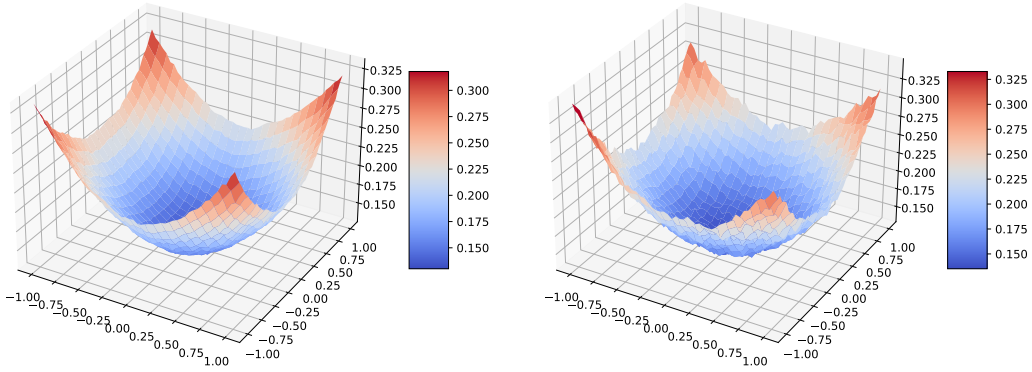
1. Though empirical results strongly support Eproxy and DPS, there is no strict mathematical proof of the upper bound of the similarity between a few-shot proxy task and a large-scale task. 2. Our experiments are limited to Computer Vision tasks. It is unknown whether the Eproxy can be extended to Natural Language Processing tasks (Vaswani et al., 2017; Hochreiter & Schmidhuber, 1997; Schuster & Paliwal, 1997; Devlin et al., 2018). 3. We didn't notice any works concurrent to our methods (According to ICLR's policy of recent work, papers appearing less than two months are considered concurrent.).



(a) Best model with the barrier.

(b) Worst model with the barrier.

Figure 7: The loss surfaces of best and worst model from NAS-Bench-101 regression task with barrier.



(a) Best model without barrier.

(b) Worst model without barrier.

Figure 8: The loss surfaces of best and worst model from NAS-Bench-101 regression task without barrier.