Generic Neural Architecture Search via Regression

Anonymous Author(s)
Affiliation
Address
email

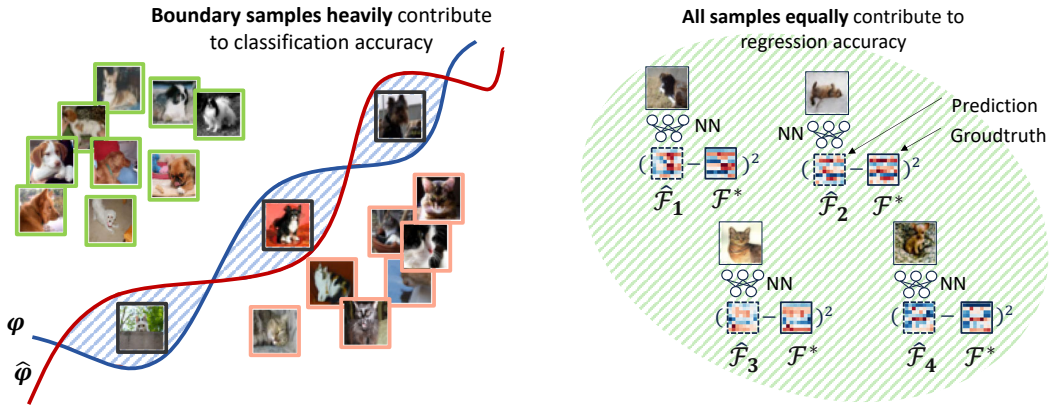
Abstract

Most existing neural architecture search (NAS) algorithms are dedicated to the downstream tasks, e.g., image classification in computer vision. However, extensive experiments have shown that, prominent neural architectures, such as ResNet in computer vision and LSTM in natural language processing, are generally good at extracting patterns from the input data and perform well on different downstream tasks. These observations inspire us to ask: Is it necessary to use the performance of specific downstream tasks to evaluate and search for good neural architectures? Can we perform NAS effectively and efficiently while being agnostic to the downstream task? In this work, we attempt to affirmatively answer the above two questions and improve the state-of-the-art NAS solution by proposing a novel and generic NAS framework, termed Generic NAS (GenNAS). GenNAS does not use task-specific labels but instead adopts *regression* on a set of manually designed synthetic signal bases for architecture evaluation. Such a self-supervised regression task can effectively evaluate the intrinsic power of an architecture to capture and transform the input signal patterns, and allow more sufficient usage of training samples. Extensive experiments across 13 CNN search spaces and one NLP space demonstrate the remarkable efficiency of GenNAS using regression, in terms of both evaluating the neural architectures (quantified by the ranking correlation Spearman's ρ between the approximated performances and the downstream task performances) and the convergence speed for training (within a few seconds). For example, on NAS-Bench-101, GenNAS achieves 0.85 ρ while the existing efficient methods only achieve 0.38. We then propose an automatic task search to optimize the combination of synthetic signals using limited downstream-task-specific labels, further improving the performance of GenNAS. We also thoroughly evaluate GenNAS's generality and end-to-end NAS performance on all search spaces, which outperforms almost all existing works with significant speedup. For example, on NASBench-201, GenNAS achieves near-optimal results with 30$\times$ speedup compared to GDAS.

1 Introduction

Most existing neural architecture search (NAS) approaches aim to find top-performing architectures on a specific downstream task, such as the prevailing image classification task in computer vision. They either directly search on the target task using the target dataset (e.g., classification on CIFAR-10 [1, 2]), or search on a *proxy* dataset and then transfer to the target one (e.g. CIFAR-10 to ImageNet) [3, 4]. However, extensive experiments show that prominent neural architectures are generally good at extracting patterns from the input data and perform well to different downstream tasks. For example, ResNet [5] being a prevailing architecture in computer vision, shows outstanding performance across various datasets and tasks [6, 7, 8], because of its advantageous architecture, the residual blocks. This observation motivates us to ask the first question: *Is there a generic way to search for and evaluate neural architectures without using the specific knowledge of downstream tasks?*

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.



(a) Classification

(b) Regression

Figure 1: In classification, only the samples near the boundary dominate the classification accuracy; in regression, all the samples equally contribute to the regression accuracy. Therefore, regression can fully use all the training samples and thus converging much faster.

Meanwhile, we observe that most existing NAS approaches directly use the *final classification performance* as the metric for evaluation and search, which has several major issues. First, the classification accuracy is dominated by the samples along the classification boundary while others have clear classification outcomes compared to the boundary ones (illustrated in Fig. 1a). Such phenomena can be observed in the limited number of effective support vectors in SVM [9], which also applies to neural networks because of the theory of neural tangent kernel [10]. Therefore, training a classifier needs many more samples than necessary (the indeed effective ones), causing a big waste. Second, a classifier tends to discard a lot of valued information, such as finer-grained features and spacial information, by transforming input representations into categorical labels. This observation motivates us to ask the second question: *Is there a more effective and efficient NAS approach, that can sufficiently use input samples and better capture valued information?*

To answer the two fundamental questions for NAS, in this work, we propose a **Gen**eric **N**eural **A**rchitecture **S**earch method, termed **GenNAS**. GenNAS adopts a **regression-based proxy task** using **downstream-task-agnostic synthetic signals** for network training and evaluation. It can efficiently (with near-zero training cost) and accurately *approximate* the neural architecture performance.

Insights. First, regression can fully use all the input samples since they equally contribute to the regression accuracy (as illustrated in Fig. 1b), as opposed to classification. Second, regression on properly-designed synthetic signals is essentially evaluating the *intrinsic representation power* of neural architectures, which is to capture and distinguish fundamental data patterns that are agnostic to the downstream tasks. Third, such representation power can be largely measured via *valued intermediate outputs* of a network (as we will show in the experiments), which are discarded by classification. **Approach**. First, we propose a *regression proxy task* as the supervising task to train, evaluate, and search for neural architectures (Fig. 2). The selected architectures will be eventually used for the target tasks. To the best of our knowledge, we are the first to propose self-supervised regression instead of classification for NAS. Second, we propose *unlabeled synthetic data* (e.g., sine signals and random signals) as the groundtruth (Fig. 3), to measure the neural architectures' intrinsic capability of capturing fundamental data patterns. Third, to further boost the performance of NAS, we propose a weakly-supervised automatic proxy task search with limited downstream-task groundtruth of in-search-space architectures(e.g. 20 architectures), to determine the best combination of synthetic signal bases targeting a specific search space and/or dataset (Fig. 4).

GenNAS Evaluation. The efficiency and effectiveness of NAS are dominated by *neural architecture evaluation*, which directs the search algorithm towards approximated top-performing network architectures. To quantify how accurate the evaluation is, one widely used indicator is the network performance *Ranking Correlation* [11] between the ranking of predicted and groundtruth ranking, denoted by Spearman's Rho (ρ) or Kendall's Tau (τ). The ideal ranking correlation is 1, when the approximated and groundtruth ranking are exactly the same; achieving large ρ or τ can significantly improve NAS quality [12, 13, 14]. Therefore, in the experiments (Sec. 4), on 13 CNN search spaces and one NLP space [15], we extensively evaluate GenNAS using the ranking correlation ρ it achieves,

and then evaluate its end-to-end NAS performance in finding the best architectures. Trained by the regression proxy task using only a single batch of unlabeled data within a few seconds, GenNAS can significantly outperform all existing NAS approaches on almost all the search spaces and datasets. For example, GenNAS achieves 0.87 ρ on NASBench-101 [16], while Zero-Cost NAS [17], the latest efficient proxy NAS approach, only achieves 0.38. On end-to-end NAS tasks, GenNAS generally outperforms other NAS approaches with large speedup. This implies that the insights behind GenNAS are plausible and that our proposed regression-based task-agnostic approach is generalizable.

Contributions. We summarize our contributions as follows:

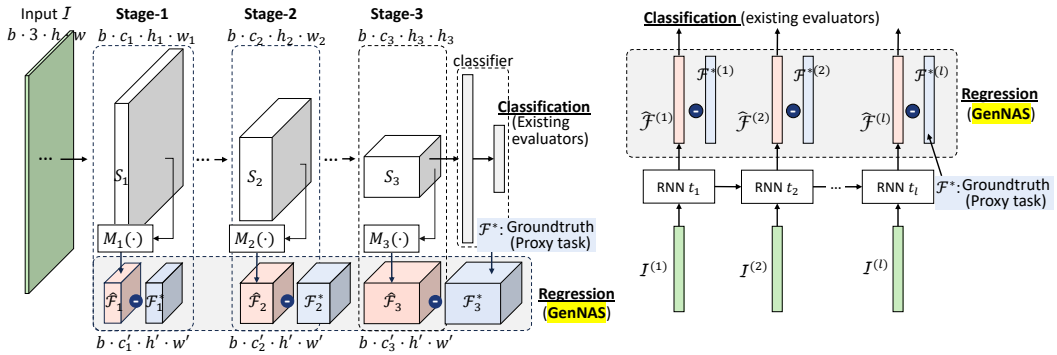
- To the best of our knowledge, GenNAS is the first NAS approach using regression as the self-supervised task instead of classification for neural architecture evaluation and search. It is thus agnostic to the specific downstream tasks, and can significantly improve the training and evaluation efficiency by fully utilizing only a handful of unlabeled data.
- GenNAS uses synthetic signal bases as the proxy task to measure the intrinsic capability of the networks that captures fundamental signal patterns. Using unlabeled synthetic data for regression, GenNAS can find the generic task-agnostic top-performing networks and can be adopted for any new search spaces with zero effort.
- An automated proxy task search to further improve GenNAS performance.
- Thorough experiments show that GenNAS outperforms existing NAS approaches by large margins in terms of ranking correlation with near-zero training cost, across 13 CNN and one NLP space *without* proxy task search. GenNAS also achieves state-of-the-art performance for end-to-end NAS with orders of magnitude of speedup over conventional methods.
- With proxy task search being optional, GenNAS is fine-tuning-free, highly efficient, and can be easily implemented on a single customer-level GPU.

2 Related Work

NAS Evaluation. Network architecture evaluation is critical in guiding the search algorithms of NAS by identifying the top-performing architectures, which is also a challenging task with intensive research interests. Early NAS works evaluated the networks by training from scratch with tremendous computation and time cost [3, 18]. To expedite, weight-sharing among the subnets sampled from a supernet is widely adopted [4, 14, 19, 20]. However, because of the poor correlation between the weight-sharing and the final performance ranking, weight-sharing NAS can easily fail even in simple search spaces [21, 22]. Yu et al. [23] further pointed out that without accurate NAS evaluation, NAS runs in a near-random fashion. Recently, zero-cost NAS methods [24, 17] have been proposed, which score the networks using the initial parameters with only one forward and backward propagation. Though achieving significant speed up, they perform poorly on identifying high-performance architectures in large search spaces such as NASBench-101. To detach the time-consuming network evaluation from NAS, several benchmarks are developed with fully-trained neural networks within the NAS search spaces [25, 16, 22, 15], so that researchers can assess the search algorithms alone in the playground.

NAS Transferability. To improve search efficiency, proxy tasks are widely used, on which the architectures are searched and then transferred to the target dataset and task. For example, the CIFAR-10 classification dataset seems to be a good proxy for ImageNet [3, 4]. Kornblith et al. [26] studied the transferability of 16 classification networks on 12 image classification datasets. NASBench-201 [22] evaluated the ranking correlations across three popular datasets with 15625 architectures, and Liu et al. [27] studied the architecture transferability across supervised and unsupervised tasks. However, even training on a downsized proxy dataset is still inefficient (e.g. a few epochs of full training is necessary [27]). By contrast, GenNAS significantly improves the efficiency by using a single batch of data while maintains extremely good generalizability across different search spaces and datasets.

Self-supervised Learning. Self-supervised learning is a form of unsupervised learning, where the neural architectures are trained with automatically generated labels to gain a good degree of comprehension or understanding [28, 29, 30, 31, 27]. Liu et al. [27] recently proposed three unlabeled classification proxy tasks, including rotation prediction, colorization, and solving jigsaw puzzles, for neural network evaluation. Though promising, this approach did not explain why such manually designed proxy tasks are beneficial, and still used classification for training with the entire dataset. By contrast, GenNAS uses regression with only a single batch of synthetic data.



(a) CNN regression (b) RNN regression

Figure 2: Regression architectures on CNNs and RNNs. (a) On CNNs, we remove the final classifier and extract multiple stages of intermediate valued information for training. (b) On RNNs, we construct a many-to-many regression task, where the input and output tensors have the same size.

3 Proposed GenNAS

In Section 3.1, we introduce the main concepts of task-agnostic GenNAS: 1) the proposed regression proxy task for both CNN architectures and recurrent neural network (RNN) architectures; 2) the synthetic signal bases used for representing the fundamental data patterns as the proxy task. In Section 3.2, we introduce the automated proxy task search.

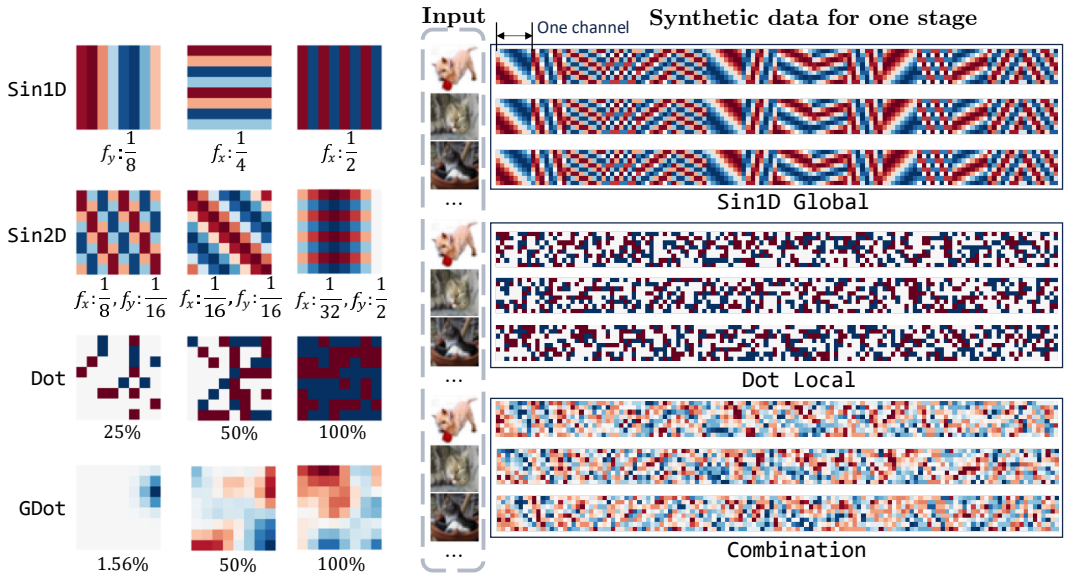
3.1 GenNAS

3.1.1 Regression Architectures

Training with unlabeled regression is the key that GenNAS being agnostic to the downstream tasks. Based on the insights discussed in Section 1, the *principle* of designing the regression architecture is to *fully utilize the abundant intermediate information* rather than the final classifier.

Regression on CNNs. Empirical studies show that CNNs learn fine-grained high-frequency spatial details in the early layers and produce the semantic features in the late layers [32]. Following the above principle, we construct a Fully Convolutional Network (FCN) [33] as shown in Fig. 2a, by removing the final classifier and extracting *multiple stages* of the intermediate feature maps. We consider N stages. **Inputs**. The inputs to the FCN are unlabeled real images, shaped as a tensor $\mathcal{I} \in \mathbb{R}^{b \times 3 \times h \times w}$, where b is the batch size, and h and w are the input image size. **Outputs**. From each stage i ($1 \leq i \leq N$) of the FCN, we first extract a feature map tensor, denoted by $\mathcal{F}_i \in \mathbb{R}^{b \times c_i \times h_i \times w_i}$, and reshape it as $\hat{\mathcal{F}}_i \in \mathbb{R}^{b \times c_i' \times h' \times w'}$ through a convolutional layer M_i by $\hat{\mathcal{F}}_i = M_i(\mathcal{F}_i)$ (with downsampling if $w_i > w'$ or $h_i > h'$). The outputs are the tensors $\hat{\mathcal{F}}_i$, which encapsulate the extracted signal patterns from different stages. **Groundtruth**. We then construct a synthetic signal tensor for each stage by combining synthetic signal bases (more details in Section 3.1.2), denoted by $\mathcal{F}_i^*$, which serves as part of the *proxy task*. We compare $\hat{\mathcal{F}}_i$ with $\mathcal{F}_i^*$ for training and evaluating the neural architectures. During training, we use MSE loss defined as $\mathcal{L} = \sum_{i=1}^{N} \mathbf{E}[(\mathcal{F}_i^* - \hat{\mathcal{F}}_i)^2]$; during validation, we adjust each stage's output importance as $\mathcal{L} = \sum_{i=1}^{N} \frac{1}{2^{N-i}} \mathbf{E}[(\mathcal{F}_i^* - \hat{\mathcal{F}}_i)^2]$ since the deeper feature map tensors are more related to the downstream task's performance. The detailed configurations of N, h', w', and c_i' will be given in the experiments.

Regression on RNNs. The proposed regression proxy task can be similarly applied to NLP tasks using RNNs. Most existing NLP models use a sequence of word-classifiers as the final outputs, whose evaluations are thus based on the word classification accuracy [34, 35, 36]. Following the same principle for CNNs, we design a many-to-many regression task for RNNs as shown in Fig. 2b. Instead of using the final word-classifier's output, we extract the output tensor of the intermediate layer before it. **Inputs**. For a general RNN model, the input is a random tensor $\mathcal{I} \in \mathbb{R}^{l \times b \times d}$, where l is the sequence length, b is the batch size, and d is the length of the input and output word vector. Given total l RNN stages, the input to each stage is one slice of the tensor $\mathcal{I}$, denoted by $\mathcal{I}^{(i)} \in \mathbb{R}^{b \times d}$, $1 \leq i \leq l$. **Outputs**. The output is $\hat{\mathcal{F}} \in \mathbb{R}^{l \times b \times d}$, where each slice for an RNN stage is $\hat{\mathcal{F}}^{(i)} \in \mathbb{R}^{b \times d}$. **Groundtruth**. Similar to CNN regression, we generate a synthetic signal tensor $\mathcal{F}^*$ as the proxy task groundtruth.



(a) Examples of signal bases

(b) Examples of synthetic signal tensors

Figure 3: (a) Examples of synthetic signal bases (2D feature maps). (b) Examples of the synthetic signal tensors by stacking 2D feature maps along the channel dimension for CNN architectures.

3.1.2 Synthetic Signal Bases

The proxy task for regression aims to capture the task-agnostic intrinsic learning capability of the neural architectures, i.e., representing various fundamental data patterns. For example, good CNNs must be able to learn different frequency signals to capture image features [32]. Here, we design four types of synthetic signal basis: (1) 1-D frequency basis (`Sin1D`); (2) 2-D frequency basis (`Sin2D`); (3) Spatial basis, `Dot` and `GDot`; (4) Resized input signal (`Resize`). `Sin1D` and `Sin2D` represent the frequency information, `Dot` and `GDot` represent the spatial information, and `Resize` represents a simple scaling task for CNNs to be scale-invariant. The combinations of these signal bases, especially with different sine frequencies, can represent a wide range of complicated signals [37]. If a network architecture is good at learning such signal basis and their simple combinations, it is more likely to be able to capture realistic signals representing different downstream tasks.

Fig.3a depicts examples of synthetic signal bases, where each is a 2D signal feature map. `Sin1D` is generated by $\sin(2\pi f x + \phi)$ or $\sin(2\pi f y + \phi)$, and `Sin2D` is generated by $\sin(2\pi f_x x + 2\pi f_y y + \phi)$, where x and y represent horizontal and vertical indices of pixels. `Dot` is generated according to biased Rademacher distribution [38] by randomly setting $k\%$ pixels to ± 1. `GDot` is generated by applying a Gaussian filter with $\sigma = 1$ on `Dot` and normalizing between ± 1. The synthetic signal tensor $\mathcal{F}^*$ (the proxy task groundtruth) is constructed by stacking the 2D signal feature maps, along the channel dimension (CNNs) or the batch dimension (for RNNs). Fig.3b shows examples of $\mathcal{F}^*$ for CNN architectures. We consider two methods for stacking: `global` and `local`. The `global` setting means the synthetic signal tensors are the same for all the inputs within the batch, as the `Sin1D Global` in Fig. 3b, aiming to test the network's ability to capture invariant features with different inputs; the `local` setting uses different synthetic signal tensors for different inputs, as the `Dot Local` in Fig.3, aiming to test the network's ability to distinguish between images. For CNN architectures, the real images are only used by `resize`, and both `global` and `local` settings are used. For RNN architectures, we only use synthetic signals and the `local` setting.

3.2 Proxy Task Search

While the synthetic signals express the generic features, the importance of the features for specific applications, NAS search spaces, and datasets may be different. Therefore, we further propose a weakly-supervised proxy task search, to automatically find the best synthetic signal tensor, i.e., the best combination of synthetic signal bases. We define the *proxy task search space* as the parameters when generating the synthetic signal tensors, which will be introduced shortly. As illustrated in Fig. 4, first, we randomly sample a small subset (e.g., 20) of the neural architectures in the NAS search space and obtain their groundtruth ranking on the target task (e.g., image classification). We then train the

Table 1: Ranking correlation (Spearman's ρ) analysis of different synthetic signals on NASBench-101.

Stage	Sin1D			Sin2D			Dot		GDot		Resize	Zero
	L	M	H	L	M	H	50%	100%	50%	100%		
S_1	0.13	0.43	**0.64**	0.14	0.53	0.63	0.55	0.62	0.18	0.16	0.56	0.17
S_2	0.03	0.52	**0.79**	0.05	0.73	0.72	0.64	0.69	0.03	0.02	0.73	0.18
S_3	0.08	0.77	0.80	0.23	0.78	0.72	0.76	**0.81**	0.16	0.17	0.80	0.22
GenNAS-combo:: 0.85												

sampled networks using different proxy tasks and calculate the performance ranking correlation ρ of the proxy and the target task. We use the regularized tournament selection evolutionary algorithm [18] to search for the task that results in the largest ρ, where ρ is the fitness function.

Proxy Task Search Space. We consider the following parameters as the proxy task search space. (1) Noise. We add noise to the input data following the distribution of parameterized Gaussian or uniform distribution. (2) The number of channels for each synthetic signal tensor (the c_i of $\mathcal{F}_i^* \in \mathbb{R}^{b\times c_i\times h'\times w'}$) can be adjusted. (3) Signal parameters, such as the frequency f and phase ϕ in `Sin`, can be adjusted. (4) Feature combination. Each synthetic signal tensor uses either `local` or `global`, and different tensors can be selected and summed up. Detailed parameters can be found in the supplemental material.

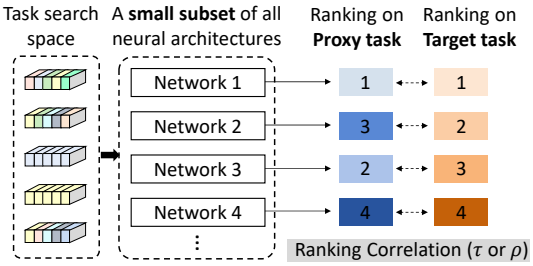


Figure 4: Proxy task search.

4 Experiments

We perform the following evaluations for GenNAS. First, in Section 4.1, we analyze the effectiveness of the synthetic signal bases and manually construct two sets of synthetic signal tensors as the baseline proxy tasks in the following evaluations. Second, in Section 4.2, we extensively evaluate the effectiveness and efficiency of the proposed regression approach by applying it on 13 CNN search spaces and one NLP search space. To show the *true power of regression*, we use manually designed proxy tasks *without* task search. GenNAS outperforms almost all existing efficient NAS approaches, achieving much higher ranking correlations (Spearman ρ) within near-zero training cost. Third, in Section 4.3, we first evaluate the effectiveness of the proxy task search, and then demonstrate the remarkable generalizability by applying one searched task to all NAS search spaces with no change. Fourth, in Section 4.4, we evaluate GenNAS on the end-to-end NAS tasks, where GenNAS also excels over the existing works with significant speedup.

Experiment Setup. We consider 13 CNN NAS search spaces including NASBench-101 [16], NASBench-201 [22], Network Design Spaces (NDS) [39], and one NLP search space, NASBench-NLP [15]. All the training is conducted using only one batch of data with batch size 16 for 100 iterations. Details of NAS search spaces and experiment settings are in the supplemental material.

4.1 Effectiveness of Synthetic Signals

The synthetic signal analysis is performed on NASBench-101 using CIFAR-10 dataset. From the whole NAS search space, 500 network architectures are randomly sampled with a known performance ranking provided by NASBench-101. We train the 500 networks using different synthetic signal tensors and calculate their ranking correlations with respect to the groundtruth ranking.

Using the CNN architecture discussed in Section 3.1.1, we consider three stages, S_1 to S_3 for $N = 3$; the number of channels is 64 for each stage. For `Sin1D` and `Sin2D`, 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) $f \in (0.375, 0.5)$. Within each frequency range, 10 signals are generated using uniformly sampled frequencies. For `Dot` and `GDot`, we randomly set 50% and 100% pixels to ± 1 on the zeroized feature maps.

The results of ranking correlations are shown in Table 1. The three stages are evaluated independently and then used together. Among the three stages, `Sin1D` and `Sin2D` within medium and high frequency work better across S_1 and S_2, while the high frequency `Dot` and `resize` work better in S_3. By contrast, low frequency signals, such as `GDot`, `Sin1D`-L, `Sin2D`-L, and the extreme case `zero` tensors, result in low ranking correlations, which we attribute to their poor distinguishing ability. We also

Table 2: GenNAS ranking correlation evaluation using the correlation Spearman's ρ. **GenNAS-single** and **GenNAS-combo** use a single or a combination of synthetic signals that are manually designed *without proxy task search*. **GenNAS search-N, -D, -R** mean the proxy task is searched on NASBench-101, NDS DARTS design space, and NDS ResNet design space, respectively.

NASBench-101

Dataset	NASWOT [24]	synflow [17]	jig ep5	jig ep10	rot ep5	rot ep10	col ep5	col ep10	cls ep5	cls ep10	GenNAS single	GenNAS combo	GenNAS search-N
CIFAR-10	0.34	0.38	0.70	0.69	0.73	0.85	0.64	0.71	0.78	0.81	0.81	0.85	**0.87**
ImgNet	0.21	0.09	0.67	0.72	0.76	**0.82**	0.54	0.67	0.68	0.79	0.73	0.73	0.73

NASBench-201

Dataset	NASWOT	synflow	jacob_cov	snip	cls@ep10 (40×)	vote	EcoNAS (60×)	GenNAS single	GenNAS combo	GenNAS search-N
CIFAR-10	0.79	0.72	0.76	0.57	0.75	0.81	0.81	0.77	0.87	**0.90**
CIFAR-100	0.81	0.76	0.70	0.61	0.75	0.83	0.75	0.69	0.82	**0.84**
ImgNet16	0.78	0.73	0.73	0.59	0.68	0.81	0.77	0.70	0.81	**0.87**

Neural Design Spaces

Dataset	NAS-Space	NASWOT	synflow	cls@ep10 (40×)	GenNAS single	GenNAS combo	GenNAS search-N	GenNAS search-D	GenNAS search-R
CIFAR-10	DARTS	0.65	0.41	0.63	0.43	0.68	0.71	**0.86**	0.82
	DARTS(f)	0.31	0.09	**0.82**	0.51	0.59	0.53	0.58	0.52
	Amoeba	0.33	0.06	0.67	0.52	0.64	0.68	**0.78**	0.72
	ENAS	0.55	0.19	0.66	0.56	0.70	0.67	**0.82**	0.78
	ENAS-f	0.43	0.26	**0.86**	0.65	0.65	0.67	0.73	0.67
	NASNet	0.40	0.00	0.64	0.56	0.66	0.65	**0.77**	0.71
	PNAS	0.51	0.26	0.50	0.32	0.58	0.59	**0.76**	0.71
	PNAS-f	0.10	0.32	**0.85**	0.45	0.48	0.56	0.55	0.47
	ResNet	0.26	0.22	0.65	0.34	0.52	0.55	0.54	**0.83**
	ResNeXt-A	0.65	0.48	0.86	0.57	0.61	0.80	0.63	**0.84**
	ResNeXt-B	0.60	0.60	0.66	0.26	0.30	0.53	0.55	**0.71**
ImageNet	DARTS	0.66	0.21	–	0.61	0.75	0.70	**0.84**	0.55
	DARTS(f)	0.20	0.37	–	0.68	0.69	0.67	**0.69**	0.59
	Amoeba	0.42	0.25	–	0.63	0.72	0.73	**0.80**	0.67
	ENAS	0.69	0.17	–	0.59	0.70	0.58	**0.81**	**0.65**
	NASNet	0.51	0.01	–	0.52	0.59	0.52	**0.70**	**0.61**
	PNAS	0.60	0.14	–	0.28	0.39	0.45	**0.62**	0.40
	ResNeXt-A	0.72	0.42	–	0.80	0.84	0.75	0.62	**0.87**
	ResNeXt-B	0.63	0.31	–	0.71	**0.79**	0.51	0.60	0.64

NASBench-NLP

Dataset	grad_norm	snip	grasp	synflow	jacob_cov	ppl@ep3 (192×)	GenNAS single	GenNAS combo	GenNAS search
PTB	0.21	0.19	0.16	0.34	0.56	0.79	0.73	0.74	**0.81**

observe that the best task in S_3 (0.81) achieves higher ρ than S_1 (0.64) and S_2 (0.79), which is consistent with the intuition that the features learned in deeper stages are more impactful for final network performance. When all three stages are used, where each stage uses its top-3 signal bases, the ranking correlation can achieve 0.85, higher than the individual stages. This supports our assumption in Section 3.1.1 that utilizing more intermediate information of a network is beneficial. From this analysis, we choose two top-performing baseline proxy tasks to be used in the following evaluations for demonstrating the effectiveness of regression: **GenNAS-single** – the best proxy task with a single signal tensor `Dot%100` used only in S_3, and **GenNAS-combo** – the combination of the three top-performing tasks in three stages.

4.2 Effectiveness and Efficiency of Regression without Proxy Task Search

To quantify how well the proposed regression can approximate the neural architecture performance with only one batch of data within seconds, we use the ranking correlation, Spearman's ρ, as the metric [17, 27, 21]. We use the two baselines proxy tasks manually designed in Section 4.1, **GenNAS-single** and **GenNAS-combo**, to show the *true power of self-supervised regression* without proxy task search. Hence, GenNAS is truly generic and can be adopted for any new search spaces. The evaluation is extensively conducted on 13 CNN search spaces and 1 NLP search space, and the results are summarized in Table 2.

On NASBench-101, GenNAS is compared with zero-cost NAS [17, 24] and the latest classification based approaches [27]. Specifically, NASWOT [24] is a zero-training approach that predicts a network's trained accuracy from its initial state by examining the overlap of activations between datapoints. Zero-Cost [17] uses proxies such as synflow to evaluate the networks, where the synflow loss is the product of all parameters in the network and has the best reported performance in the paper. Liu et al. [27] used three unsupervised classification training proxies, namely rotation prediction (rot), colorization (col), and solving jigsaw puzzles (jig), and one supervised classification proxy (cls).

We report their results after 5 and 10 epochs (ep5 and ep10) for each proxy. The results show that GenNAS-single and GenNAS-combo achieve 0.81 and 0.85 ρ on CIFAR-10, and achive 0.73 on ImageNet, respectively, much higher than NASWOT and synflow. It is also comparable and even higher comparing with the classification proxies, cls@ep5 and cls@ep10. Notably, the classification proxies need to train for 5 or 10 epochs using *all training data*, while GenNAS requires only a few seconds, more than $40\times$ faster. On NASBench-201, we further compare with vote [17] and EcoNAS [12] in addition to NASWOT and synflow. EcoNAS is a recently proposed reduced-training proxy NAS approach. vote [17] adopts the majority vote between three zero-training proxies including synflow, jacob_cov, and snip. Clearly, GenNAS-combo outperforms all these methods regarding ranking correlation, and is also $60\times$ faster than EcoNAS and $40\times$ faster than cls@ep10. On Neural Design Spaces, we evaluate GenNAS on both CIFAR-10 and ImageNet datasets. Comparing with NASWOT and synflow, GenNAS-single and GenNAS-combo achieve higher ρ in almost all cases. Also, synflow performs poorly on most of the NDS search spaces especially on ImageNet dataset, while GenNAS achieves even higher ρ. Extending to NLP search space, NASBench-NLP, GenNAS-single and GenNAS-combo achieve 0.73 and 0.74 ρ, respectively, surpassing the best zero-proxy method (0.56). Comparing with the ppl@ep3, the architectures trained on PTB [40] dataset after three epochs, GenNAS is $192\times$ faster.

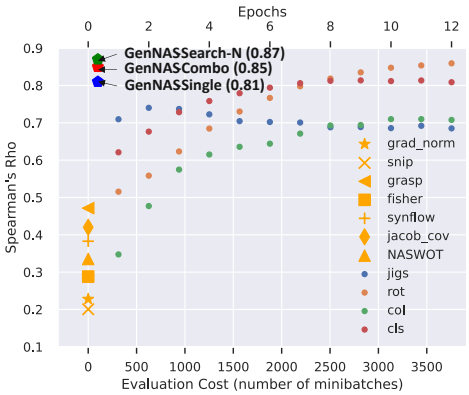


Figure 5: The effectiveness of regression-based proxy task. GenNAS significantly outperforms all the existing NAS evaluation approaches regarding ranking correlation, with near-zero training cost.

Fig. 5 visualizes the comparisons between GenNAS and existing NAS approaches on NASBench-101, CIFAR-10. Clearly, regression-based GenNAS (single, combo) significantly outperforms the existing NAS with near-zero training cost, showing remarkable effectiveness and high efficiency.

4.3 Effectiveness of Proxy Task Search and Transferability

Effectiveness of Proxy Task Search. While the *unsearched* proxy tasks can already significantly outperform all existing approaches (shown in Section 4.2), we demonstrate that the proxy task search described in Section 3.2 can further improve the ranking correlation. We adopt the regularized evolutionary algorithm [18]. The population size is 50; the tournament sample size is 10; the search runs 400 iterations. We randomly select 20 architectures with groundtruth for calculating the ρ. More settings can be found in the supplemental material. Fig. 6 shows the search results averaged from 10 runs with different seeds. It shows that the regularized evolutionary algorithm is more effective comparing with random search, where the correlations of 20 architectures are 0.86 ± 0.02 and 0.82 ± 0.01, respectively.

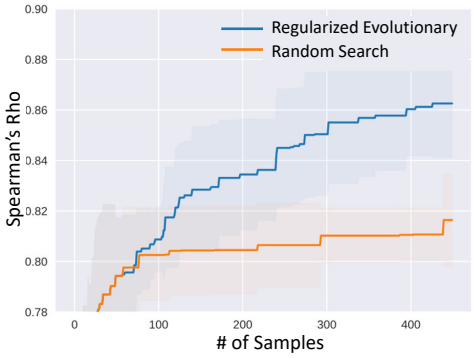


Figure 6: Proxy task search.

In the following experiments, we evaluate three searched proxy tasks, denoted by **GenNAS search-N**, **-D**, and **-R**, meaning that the task is searched on NASBench-101, NDS DARTS search space, and NDS ResNet search space, respectively. We study the performance and transferability of the searched tasks on all NAS search spaces.

GenNAS with Searched/Transferred Proxy Task. The performance of GenNAS-search-N, -D, and -R proxy tasks is shown in Table 2. First, in most cases, proxy task search improves the ranking correlation. For example, in NDS, using the proxy task searched on DARTS space (search-D) outperforms other GenNAS settings on DARTS-like spaces, while using proxy task search-R on ResNet-like spaces outperforms others as well. In NASBench-NLP, the proxy task search can push the

Table 3: GenNAS end-to-end NAS results comparing with the state-of-the-art NAS approaches, showing test accuracy (%) on different NAS-spaces and datasets. $\star$ denotes a method is replicated with the same RE algorithm in Section 4.4 to achieve a fair comparison.

NASBench-101(CIFAR-10)								
Optimal	NASWOT*	synflow*	Halfway*	**GenNAS-N**				
94.32	93.30±0.002	91.31±0.02	93.28±0.002	**93.92±0.004**				
NASBench-201								
Dataset	Optimal	RSPS	DARTS-V2	GDAS	SETN	ENAS	NASWOT	**GenNAS-N**
CIFAR-10	94.37	84.07±3.61	54.30±0.00	93.61±0.09	87.64±0.00	53.89±0.58	92.96±0.80	**94.18±0.10**
CIFAR-100	73.49	58.33±4.34	15.61±0.00	70.61±0.26	56.87±7.77	15.61±0.00	70.03±1.16	**72.56±0.74**
ImgNet16	47.31	26.28±3.09	16.32±0.00	41.71±0.98	32.52±0.21	14.84±2.10	44.43±2.07	**45.59±0.54**
GPU hours		2.2	9.9	8.8	9.5	3.9	0.1	0.3
Neural Design Spaces (CIFAR-10)								
NAS-Space	Optimal	NASWOT*	synflow*	cls@ep10*	**GenNAS-N**	**GenNAS-R**	**GenNAS-D**	
ResNet	95.30	92.81±0.10	93.52±0.31	94.51±0.20	94.48±0.24	94.63±0.23	**94.77±0.13**	
ResNeXt-A	94.99	93.39±0.67	94.05±0.48	94.24± 0.22	94.25±0.21	94.12±0.20	**94.37±0.14**	
ResNeXt-B	95.12	93.56±0.33	93.65±0.64	94.33±0.26	**94.29±0.24**	94.26±0.35	94.23±0.32	

ranking correlation to 0.81, surpassing the ppl@ep3 (0.79). Such results demonstrate the effectiveness of our proposed proxy task search. Second, the searched proxy task shows great transferability: the proxy task searched on NASBench-101 (search-N) generally works well for other search spaces, i.e., NASBench-201, NDS, and NASBench-NLP. This further reinforces that the fundamental principles for top-performing neural architectures are similar across different tasks and datasets. Fig. 5 also includes the performance of GenNAS-search-N comparing with others.

4.4 GenNAS for End-to-End NAS

Finally, we evaluate GenNAS on the end-to-end NAS tasks, aiming to find the best neural architectures within the search space. Table 3 summarizes the comparisons with the state-of-the-art NAS approaches, including previously used NASWOT, synflow, cls@ep10, and additionally Halfway [16], RSPS [14], DARTS-V1 [4], DARTS-V2, GDAS [41], SETN [42], and ENAS [43]. Halfway is the NASBench-101-released result using half of the total epochs for network training. In all the searches during NAS, we do not use any tricks such as warmup selection [17] or groundtruth query to compensate the low rank correlations. We fairly use a simple yet effective regularized evolutionary algorithm [18] and adopt the proposed regression loss as the fitness function. The population size is 50 and the tournament sample size is 10 with 400 iterations. On NASBench-101, GenNAS finds better architectures than NASWOT and Halfway while being up to $200\times$ faster. On NASBench-201, GenNAS finds better architectures than the state-of-the-art GDAS within 0.3 GPU hours, being $30\times$ faster. Note that GenNAS uses the proxy task searched on NASBench-101 and transferred to NASBench-201, demonstrating remarkable transferability. On Neural Design Spaces, GenNAS finds better architectures than the cls@ep10 using labeled classification while being $40\times$ faster. On NASBench-NLP, GenNAS finds architectures that achieve 0.246 (the lower the better) average final regret score r, outperforming the ppl@ep3 (0.268) with $192\times$ speed up. The regret score r at the moment t is $r(t) = L(t) - L^\star$, where $L(t)$ is the testing log perplexity of the best found architecture according to the prediction by the moment, and $L^\star = 4.36$ is the lowest testing log perplexity in the whole dataset achieved by LSTM[34] architecture. These end-to-end NAS experiments strongly suggest that GenNAS is generically efficient and effective across different search spaces and datasets.

5 Conclusion

In this work, we proposed GenNAS, a self-supervised regression-based approach for neural architecture training, evaluation, and search. GenNAS successfully answered the two questions at the beginning. (1) GenNAS *is* a generic task-agnostic method, using synthetic signals to capture their fundamental learning capability without specific downstream task knowledge. (2) GenNAS *is* an extremely effective and efficient method using regression, fully utilizing all the training samples and better capturing valued information. We show the true power of self-supervised regression via manually designed proxy tasks that do not need to search. Extensive experiments confirmed that GenNAS is able to deliver state-of-the-art performance with near-zero search time, in terms of both ranking correlation and the end-to-end NAS tasks with great generalizability.

References

[1] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

[2] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In *International Conference on Machine Learning*, pages 2902–2911. PMLR, 2017.

[3] 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.

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

[5] 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.

[6] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[7] Mark Marsden, Kevin McGuinness, Suzanne Little, and Noel E O'Connor. Resnetcrowd: A residual deep learning architecture for crowd counting, violent behaviour detection and crowd density level classification. In *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–7. IEEE, 2017.

[8] Maithra Raghu, Chiyuan Zhang, Jon Kleinberg, and Samy Bengio. Transfusion: Understanding transfer learning for medical imaging. *arXiv preprint arXiv:1902.07208*, 2019.

[9] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

[10] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *arXiv preprint arXiv:1806.07572*, 2018.

[11] Wayne W Daniel et al. Applied nonparametric statistics. 1990.

[12] Dongzhan Zhou, Xinchi Zhou, Wenwei Zhang, Chen Change Loy, Shuai Yi, Xuesen Zhang, and Wanli Ouyang. Econas: Finding proxies for economical neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11396–11404, 2020.

[13] Xiangxiang Chu, Bo Zhang, Ruijun Xu, and Jixiang Li. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. *arXiv preprint arXiv:1907.01845*, 2019.

[14] Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. In *Uncertainty in Artificial Intelligence*, pages 367–377. PMLR, 2020.

[15] Nikita Klyuchnikov, Ilya Trofimov, Ekaterina Artemova, Mikhail Salnikov, Maxim Fedorov, and Evgeny Burnaev. Nas-bench-nlp: neural architecture search benchmark for natural language processing. *arXiv preprint arXiv:2006.07116*, 2020.

[16] 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.

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

[18] 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.

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

[20] 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.

[21] Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. Understanding and robustifying differentiable architecture search. *arXiv preprint arXiv:1909.09656*, 2019.

[22] Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. *arXiv preprint arXiv:2001.00326*, 2020.

[23] Kaicheng Yu, Christian Sciuto, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. Evaluating the search phase of neural architecture search. *arXiv preprint arXiv:1902.08142*, 2019.

[24] Joseph Mellor, Jack Turner, Amos Storkey, and Elliot J Crowley. Neural architecture search without training. *arXiv preprint arXiv:2006.04647*, 2020.

[25] Julien Siems, Lucas Zimmer, Arber Zela, Jovita Lukasik, Margret Keuper, and Frank Hutter. Nas-bench-301 and the case for surrogate benchmarks for neural architecture search. *arXiv preprint arXiv:2008.09777*, 2020.

[26] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2661–2671, 2019.

[27] 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.

[28] Longlong Jing and Yingli Tian. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[29] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. Citeseer, 2014.

[30] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pages 1422–1430, 2015.

[31] Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *Proceedings of the IEEE international conference on computer vision*, pages 2794–2802, 2015.

[32] 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.

[33] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

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

[35] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010.

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[37] Georgi P Tolstov. *Fourier series*. Courier Corporation, 2012.

[38] Stephen J Montgomery-Smith. The distribution of rademacher sums. *Proceedings of the American Mathematical Society*, 109(2):517–522, 1990.

[39] Ilija Radosavovic, Justin Johnson, Saining Xie, Wan-Yen Lo, and Piotr Dollár. On network design spaces for visual recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1882–1890, 2019.

[40] Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. 1993.

[41] 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.

[42] Xuanyi Dong and Yi Yang. One-shot neural architecture search via self-evaluated template network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3681–3690, 2019.

[43] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning*, pages 4095–4104. PMLR, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]