Generic Neural Architecture Search via Regression

Yuhong Li¹, Cong Hao², Pan Li³, Jinjun Xiong⁴, Deming Chen¹
University of Illinois at Urbana-Champaign¹, Georgia Institute of Technology²,
Purdue University³, University at Buffalo ⁴
leeyh@illinois.edu, callie.hao@ece.gatech.edu, panli@purdue.edu,
jinjun@buffalo.edu, dchen@illinois.edu

Abstract

Most existing neural architecture search (NAS) algorithms are dedicated to and evaluated by 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. In this paper, we attempt to answer two fundamental questions related to NAS. (1) Is it necessary to use the performance of specific downstream tasks to evaluate and search for good neural architectures? (2) Can we perform NAS effectively and efficiently while being agnostic to the downstream tasks? To answer these questions, we propose 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 can find near-optimal architectures within 0.3 GPU hour. Our code has been made available at: https://github.com/leeyehoo/GenNAS

1 Introduction

Most existing neural architecture search (NAS) approaches aim to find top-performing architectures on a specific downstream task, such as image classification [1, 2, 3, 4, 5], semantic segmentation [6, 7, 8], neural machine translation [9, 10, 11] or more complex tasks like hardware-software co-design [12, 13, 14, 15, 16]. They either directly search on the target task using the target dataset (e.g., classification on CIFAR-10 [2, 17]), or search on a proxy dataset and then transfer to the target one (e.g. CIFAR-10 to ImageNet) [18, 19]. 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 [19] being a prevailing architecture in computer vision, shows outstanding performance across various datasets and tasks [20, 21, 22], because of its
samples near the classification boundary heavily affect the classification accuracy. Therefore, regression is better at leveraging all training samples than classification to achieve faster convergence.

Meanwhile, we observe that most existing NAS approaches directly use the final classification performance as the metric for architecture evaluation and search, which has several major issues. First, the classification accuracy is dominated by the samples along the classification boundary, while other samples have clearer classification outcomes compared to the boundary ones (as illustrated in Fig. 1a). Such phenomena can be observed in the limited number of effective support vectors in SVM [23], which also applies to neural networks because of the theory of neural tangent kernel [24]. Therefore, discriminating performance of classifiers needs many more samples than necessary (the indeed effective ones), causing a big waste. Second, a classifier tends to discard a lot of valuable information, such as finer-grained features and spatial information, by transforming input representations into categorical labels. This observation motivates us to ask the second question: Is there a more effective way that can make more sufficient use of input samples and better capture valuable information?

To answer the two fundamental questions for NAS, in this work, we propose a Generic Neural Architecture Search 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, as opposed to classification, regression can efficiently make fully use of all the input samples, which equally contribute to the regression accuracy (Fig. 1b). 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 downstream tasks. Third, such representation power is heavily reflected in the intermediate data of a network (as we will show in the experiments), which are regretfully 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). Then, the searched architectures will be used for the target downstream tasks. To the best of our knowledge, we are the first to propose self-supervised regression proxy task instead of classification for NAS. Second, we propose to use unlabeled synthetic data (e.g., sine and random signals) as the groundtruth (Fig. 3) to measure neural architectures’ intrinsic capability of capturing fundamental data patterns. Third, to further boost NAS performance, we propose a weakly-supervised automatic proxy task search with only a handful of groundtruth architecture performance (e.g. 20 architectures), to determine the best proxy task, i.e., the combination of synthetic signal bases, targeting a specific downstream task, 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 top-performing network architectures. To quantify how accurate the evaluation is, one widely used indicator is the network performance Ranking
Correlation \cite{25} between the prediction and groundtruth ranking, defined as Spearman’s Rho (\(\rho\)) or Kendall’s Tau (\(\tau\)). The ideal ranking correlation is 1 when the approximated and groundtruth rankings are exactly the same; achieving large \(\rho\) or \(\tau\) can significantly improve NAS quality \cite{26,27,28}. Therefore, in the experiments (Sec. 4), we evaluate GenNAS using the ranking correlation factors it achieves, and then show its end-to-end NAS performance in finding the best architectures. Extensive experiments are done on 13 CNN search spaces and one NLP space \cite{29}. Trained by the regression proxy task using only a single batch of unlabeled data within a few seconds, GenNAS significantly outperforms all existing NAS approaches on almost all the search spaces and datasets. For example, GenNAS achieves 0.87 \(\rho\) on NASBench-101 \cite{30}, while Zero-Cost NAS \cite{31}, an efficient proxy NAS approach, only achieves 0.38. On end-to-end NAS, GenNAS generally outperforms others with large speedup. This implies that the insights behind GenNAS are plausible and that our proposed regression-based task-agnostic approach is generalizable across tasks, search spaces, and datasets.

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 proxy task instead of classification for neural architecture evaluation and search. It is agnostic to the specific downstream tasks and can significantly improve training and evaluation efficiency by fully utilizing only a handful of unlabeled data.
- GenNAS uses synthetic signal bases as the groundtruth to measure the intrinsic capability of networks that captures fundamental signal patterns. Using such unlabeled synthetic data in regression, GenNAS can find the generic task-agnostic top-performing networks and can apply to 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 \cite{18,1}. To expedite, weight-sharing among the subnets sampled from a supernet is widely adopted \cite{3,28,4,32,33}. However, due to the poor correlation between the weight-sharing and the final performance ranking, weight-sharing NAS can easily fail even in simple search spaces \cite{34,35}. Yu et al. \cite{36} further pointed out that without accurate evaluation, NAS runs in a near-random fashion. Recently, zero-cost NAS methods \cite{37,31,38,39} have been proposed, which score the networks using their initial parameters with only one forward and backward propagation. Despite the significant speed up, they fail to identify top-performing 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 \cite{40,30,35,29,41}, 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 target datasets and tasks. For example, the CIFAR-10 classification dataset seems to be a good proxy for ImageNet \cite{18,3}. Kornblith et al. \cite{42} studied the transferability of 16 classification networks on 12 image classification datasets. NASBench-201 \cite{35} evaluated the ranking correlations across three popular datasets with 15625 architectures. Liu et al. \cite{43} studied the architecture transferability across supervised and unsupervised tasks. Nevertheless, training on a downsized proxy dataset is still inefficient (e.g. a few epochs of full-blown training \cite{44}). In contrast, GenNAS significantly improves the efficiency by using a single batch of data while maintaining extremely good generalizability across different search spaces and datasets.

Self-supervised Learning. Self-supervised learning is a form of unsupervised learning, that the neural architectures are trained with automatically generated labels to gain a good degree of com-
Figure 2: Regression architectures on CNNs and RNNs. (a) On CNNs, we remove the final classifier and extract multiple stages of intermediate feature map for training. (b) On RNNs, we construct a many-to-many regression task, where the input and output tensors have the same size.

prehension or understanding \[44, 45, 46, 47, 43\]. Liu et al. \[43\] 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. In contrast, GenNAS uses regression with only a single batch of synthetic data.

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 using unlabeled regression is the key that GenNAS being agnostic to 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 semantic features in the late layers \[48\]. Following this principle, as shown in Fig. 2a, we construct a Fully Convolutional Network (FCN) \[49\] by removing the final classifier of a CNN, and then extract the FCN’s intermediate feature maps from multiple stages. We denote the number of stages as \(N\).

Inputs. The inputs to the FCN are unlabeled real images, shaped as a tensor \(I \in \mathbb{R}^{b \times 3 \times h \times w}\), 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 \(F_i \in \mathbb{R}^{b \times c_i \times h_i \times w_i}\), and reshape it as \(\hat{F}_i \in \mathbb{R}^{b \times c'_i \times h'_i \times w'_i}\) through a convolutional layer \(M_i\) by \(\hat{F}_i = M_i(F_i)\) (with downsampling if \(w_i > w'_i\) or \(h_i > h'_i\)). The outputs are the tensors \(\hat{F}_i\), which encapsulate the captured signal patterns from different stages. Groundtruth. We construct a synthetic signal tensor for each stage as the groundtruth, which serves as part of the proxy task. A synthetic tensor is a combination of multiple synthetic signal bases (more details in Section 3.1.2), denoted by \(F_i^*\). We compare \(\hat{F}_i\) with \(F_i^*\) for training and evaluating the neural architectures. During training, we use MSE loss defined as \(L = \sum_{i=1}^{N} E[(\hat{F}_i - F_i)^2]\); during validation, we adjust each stage’s output importance as \(L = \sum_{i=1}^{N} \frac{1}{2^i} E[(\hat{F}_i - F_i)^2]\) since the feature map tensors of later stages are more related to the downstream task’s performance. The detailed configurations of \(N, h', w',\) and \(c'_i\) are provided 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,
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 \[53\]. Here, we design four types of synthetic signal basis: (1) 1-D frequency basis \((\text{Sin1D})\); (2) 2-D frequency basis \((\text{Sin2D})\); (3) Spatial basis \((\text{Dot} \text{ and } \text{GDot})\); (4) Resized input signal \((\text{Resize})\). \text{Sin1D} \text{ and } \text{Sin2D} \text{ represent frequency information, } \text{Dot} \text{ and } \text{GDot} \text{ represent spatial information, and Resize reflects the CNN’s scale-invariant capability. The combinations of these signal bases, especially with different sine frequencies, can represent a wide range of complicated real-world signals \[54\]. If a network architecture is good at learning such signal basis and their simple combinations, it is more likely to be able to capture real-world signals from different downstream tasks.

Fig. 3 depicts examples of synthetic signal bases, where each base is a 2D signal feature map. \text{Sin1D} is generated by \(\sin(2\pi fx + \phi)\) or \(\sin(2\pi fy + \phi)\), and \text{Sin2D} is generated by \(\sin(2\pi fx + 2\pi fy + \phi)\), where \(x\) and \(y\) are pixel indices. \text{Dot} is generated according to biased Rademacher distribution \[55\] by randomly setting \(k\%\) pixels to \(\pm 1\) on zeroed feature maps. \text{GDot} is generated by applying a Gaussian filter with \(\sigma = 1\) on \text{Dot} and normalizing between \(\pm 1\). The synthetic signal tensor \(\hat{F}^*\) (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. 3 shows examples of stacked synthetic tensor \(\hat{F}^*\) for CNN architectures. Within one batch of input images, we consider two settings: global and local. The global setting means that the synthetic tensor is the same for all the inputs within the batch, as the \text{Sin2D Global} in Fig. 3, aiming to test the network’s ability to capture invariant features from different inputs; the local setting uses different synthetic signal tensors for different inputs, as the \text{Dot Local} in Fig. 3, aiming to test the network’s ability to distinguish between images. For CNNs, the real images are only used by \text{resize}, and both \text{global} and \text{local} settings are used. For RNNs, we only use synthetic signals and the \text{local} setting, because resizing natural language or time series, the typical input of RNNs, does not make as much sense as resizing images for CNNs.
3.2 Proxy Task Search

While the synthetic signals can express generic features, the importance of these features for different tasks, 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. 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 these networks using different proxy tasks and calculate the performance ranking correlation $\rho$ of the proxy and the target task. We use the regularized tournament selection evolutionary algorithm [1] to search for the task that results in the largest $\rho$, where $\rho$ 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 ($c_i$ in $F^* \in \mathbb{R}^{h \times c_i \times h' \times w'}$) can be adjusted. (3) Signal parameters, such as the frequency $f$ and phase $\phi$ in $\text{Sin}$, can be adjusted. (4) Feature combination. Each synthetic signal tensor uses either local or global, and tensors can be selected and summed up. Detailed parameters can be found in the supplemental material.

4 Experiments

We perform the following evaluations for GenNAS. First, to show the true power of regression, we use manually designed proxy tasks without task search and apply the same proxy task on all datasets and search spaces. We demonstrate that the GenNAS generally excels in all different cases with zero task-specific cost, thanks to unlabeled self-supervised regression proxy tasks. Specifically, in Section 4.1, we analyze the effectiveness of the synthetic signal bases and manually construct two sets of synthetic tensors as the baseline proxy tasks; in Section 4.2, we extensively evaluate the proposed regression approach in 13 CNN search spaces and one NLP search space. Second, in Section 4.3, we evaluate the proxy task search and demonstrate the remarkable generalizability by applying one searched task to all NAS search spaces with no change. Third, in Section 4.4, we evaluate GenNAS on end-to-end NAS tasks, which outperforms existing works with significant speedup.

**Experiment Setup.** We consider 13 CNN NAS search spaces including NASBench-101 [30], NASBench-201 [35], Network Design Spaces (NDS) [56], and one NLP search space, NASBench-NLP [29]. 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 5.1.1, 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 in $S_2$ and $S_3$, while the high frequency Dot and resize work better in $S_3$. The low frequency signals, such as GDot, Sin1D-L, Sin2D-L, and the extreme case zero tensors, result in low ranking correlations; we attribute to their poor distinguishing ability. We also observe that the
Table 1: Ranking correlation (Spearman’s ρ) analysis of different synthetic signals on NASBench-101.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Sin1D</th>
<th>Sin2D</th>
<th>Dot</th>
<th>GDot</th>
<th>Resize</th>
<th>Zero</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>L</td>
<td>M</td>
<td>H</td>
<td>L</td>
<td>M</td>
<td>H</td>
</tr>
<tr>
<td>S₁</td>
<td>0.13</td>
<td>0.43</td>
<td>0.64</td>
<td>0.14</td>
<td>0.53</td>
<td>0.63</td>
</tr>
<tr>
<td>S₂</td>
<td>0.03</td>
<td>0.52</td>
<td>0.79</td>
<td>0.05</td>
<td>0.73</td>
<td>0.72</td>
</tr>
<tr>
<td>S₃</td>
<td>0.08</td>
<td>0.77</td>
<td>0.80</td>
<td>0.23</td>
<td>0.78</td>
<td>0.72</td>
</tr>
</tbody>
</table>

GenNAS-combo: 0.85

best task in S₃ (0.81) achieves higher ρ than S₁ (0.64) and S₂ (0.79), which is consistent with the intuition that the features learned in deeper stages have more impact to the 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 proxy tasks in the following evaluations to demonstrate the effectiveness of regression: GenNAS-single – the best proxy task with a single signal tensor Dot%100 used only in S₃, 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 [31, 43, 34]. We use the two manually designed proxy tasks (GenNAS-single and GenNAS-combo) without proxy task search to demonstrate that GenNAS is generic and can be directly applied to any new search spaces with zero task-specific search efforts. 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 [31, 37] and the latest classification based approaches [43]. Specifically, NASWOT [37] is a zero-training approach that predicts a network’s trained accuracy from its initial state by examining the overlap of activations between datapoints. Abdelfattah et al. [31] proposed proxies such as synflow to evaluate the networks, where the synflow computes the summation of all the weights multiplied by their gradients and has the best reported performance in the paper. Liu et al. [43] 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 10 epochs (@ep10) for each proxy. The results show that GenNAS-single and GenNAS-combo achieve 0.81 and 0.85 ρ on CIFAR-10, and achieve 0.73 on ImageNet, respectively, much higher than NASWOT and synflow. It is also comparable and even higher compared to the classification proxies, cls@ep5 and cls@ep10. Notably, the classification proxies need to train for 10 epochs using all training data, while GenNAS requires only a few seconds, more than 40× faster. On NASBench-201, we further compare with vote [31] and EcoNAS [26]. EcoNAS is a recently proposed reduced-training proxy NAS approach. Vote [31] 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× faster than EcoNAS and 40× 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
Table 2: GenNAS ranking correlation evaluation using the correlation Spearman’s $\rho$. 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. The top-1/2/3 results of GenNAS and efficient NAS baselines are highlighted by $^1$, $^2$, $^3$ respectively for each task. The values with superscripts are obtained after task search ($^*$) or transferred ($^\dagger$) from a previous searched task. Methods like jig@ep10 which is 40x slower compared to the GenNAS in prediction are not considered as efficient ones.

### NASBench-101

<table>
<thead>
<tr>
<th>Dataset</th>
<th>NASWOT</th>
<th>syfllow</th>
<th>jigg@ep10</th>
<th>net@ep10</th>
<th>col@ep10</th>
<th>cls@ep10</th>
<th>GenNAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR-10</td>
<td>0.34</td>
<td>0.38</td>
<td>0.69</td>
<td>0.85</td>
<td>0.71</td>
<td>0.81</td>
<td>0.83 $^1$</td>
</tr>
<tr>
<td>ImgNet</td>
<td>0.21</td>
<td>0.09</td>
<td>0.72</td>
<td>0.82</td>
<td>0.67</td>
<td>0.79</td>
<td>0.65 $^3$</td>
</tr>
</tbody>
</table>

### NASBench-201

<table>
<thead>
<tr>
<th>Dataset</th>
<th>NASWOT</th>
<th>jacob_cov</th>
<th>snup</th>
<th>cls@ep10</th>
<th>vote</th>
<th>EcoNAS</th>
<th>GenNAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR-10</td>
<td>0.79 $^1$</td>
<td>0.72</td>
<td>0.76</td>
<td>0.57</td>
<td>0.75</td>
<td>0.81</td>
<td>0.81</td>
</tr>
<tr>
<td>CIFAR-100</td>
<td>0.81</td>
<td>0.76</td>
<td>0.70</td>
<td>0.61</td>
<td>0.75</td>
<td>0.83 $^1$</td>
<td>0.75</td>
</tr>
<tr>
<td>ImgNet16</td>
<td>0.78</td>
<td>0.73</td>
<td>0.73</td>
<td>0.99</td>
<td>0.68</td>
<td>0.81 $^1$</td>
<td>0.77</td>
</tr>
</tbody>
</table>

### Neural Design Spaces

<table>
<thead>
<tr>
<th>Dataset</th>
<th>NAS-Space</th>
<th>NASWOT</th>
<th>syfllow</th>
<th>cls@ep10</th>
<th>GenNAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR-10</td>
<td>DARTS</td>
<td>0.65</td>
<td>0.41</td>
<td>0.63</td>
<td>0.43</td>
</tr>
<tr>
<td>DARTS-f</td>
<td>0.31</td>
<td>0.09</td>
<td>0.82</td>
<td>0.51</td>
<td>0.59 $^3$</td>
</tr>
<tr>
<td>Amoeba</td>
<td>0.33</td>
<td>0.06</td>
<td>0.67</td>
<td>0.52</td>
<td>0.64</td>
</tr>
<tr>
<td>ENAS</td>
<td>0.55</td>
<td>0.19</td>
<td>0.66</td>
<td>0.56</td>
<td>0.70 $^3$</td>
</tr>
<tr>
<td>ENAS-f</td>
<td>0.43</td>
<td>0.26</td>
<td>0.86</td>
<td>0.65</td>
<td>0.65</td>
</tr>
<tr>
<td>NASNet</td>
<td>0.40</td>
<td>0.00</td>
<td>0.64</td>
<td>0.56</td>
<td>0.66 $^3$</td>
</tr>
<tr>
<td>PNAS</td>
<td>0.51</td>
<td>0.26</td>
<td>0.50</td>
<td>0.32</td>
<td>0.58</td>
</tr>
<tr>
<td>PNAS-f</td>
<td>0.10</td>
<td>0.32</td>
<td>0.85</td>
<td>0.45</td>
<td>0.48 $^3$</td>
</tr>
<tr>
<td>ResNet</td>
<td>0.26</td>
<td>0.22</td>
<td>0.65</td>
<td>0.34</td>
<td>0.52</td>
</tr>
<tr>
<td>ResNeXt-A</td>
<td>0.65 $^1$</td>
<td>0.48</td>
<td>0.86</td>
<td>0.57</td>
<td>0.61</td>
</tr>
<tr>
<td>ResNeXt-B</td>
<td>0.60 $^3$</td>
<td>0.60 $^3$</td>
<td>0.66</td>
<td>0.26</td>
<td>0.30</td>
</tr>
<tr>
<td>ImageNet</td>
<td>DARTS</td>
<td>0.66</td>
<td>0.21</td>
<td>--</td>
<td>0.61</td>
</tr>
<tr>
<td>DARTS-f</td>
<td>0.20</td>
<td>0.37</td>
<td>--</td>
<td>0.68</td>
<td>0.69 $^3$</td>
</tr>
<tr>
<td>Amoeba</td>
<td>0.42</td>
<td>0.25</td>
<td>--</td>
<td>0.63</td>
<td>0.72 $^3$</td>
</tr>
<tr>
<td>ENAS</td>
<td>0.69 $^1$</td>
<td>0.17</td>
<td>--</td>
<td>0.59</td>
<td>0.70 $^3$</td>
</tr>
<tr>
<td>NASNet</td>
<td>0.51</td>
<td>0.01</td>
<td>--</td>
<td>0.52</td>
<td>0.59</td>
</tr>
<tr>
<td>PNAS</td>
<td>0.60 $^1$</td>
<td>0.14</td>
<td>--</td>
<td>0.28</td>
<td>0.39</td>
</tr>
<tr>
<td>ResNeXt-A</td>
<td>0.72</td>
<td>0.42</td>
<td>--</td>
<td>0.80 $^1$</td>
<td>0.64 $^3$</td>
</tr>
<tr>
<td>ResNeXt-B</td>
<td>0.63</td>
<td>0.31</td>
<td>--</td>
<td>0.71 $^1$</td>
<td>0.79 $^3$</td>
</tr>
</tbody>
</table>

### NASBench-NLP

<table>
<thead>
<tr>
<th>Dataset</th>
<th>grad_norm</th>
<th>snap</th>
<th>grasp</th>
<th>jacob_cov</th>
<th>ppl@ep3</th>
<th>GenNAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTB</td>
<td>0.21</td>
<td>0.19</td>
<td>0.16</td>
<td>0.34</td>
<td>0.56 $^1$</td>
<td>0.79</td>
</tr>
</tbody>
</table>

zero-proxy method (0.56). Comparing with the ppl@ep3, the architectures trained on PTB dataset after three epochs, GenNAS is 192x faster in prediction.

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 unssearched 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 [1]. 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 $\rho$. 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 $0.86 \pm 0.02$ and $0.82 \pm 0.01$, respectively.
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 space, and NDS ResNet search space, respectively. We study the performance and transferability of the searched architectures on all NAS search spaces. Proxy task search is done on a single GPU GTX 1080Ti. On NASBench-101 (GenNAS-N), ResNeXt (GenNAS-R), and DARTS (GenNAS-D), the search time is 5.75, 4, and 12.25 GPU hours, respectively. Once the proxy tasks are searched, it can be used to evaluate any architectures in the target search space and can be transferred to other 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 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 emphasizes that the fundamental principles for top-performing neural architectures are similar across different tasks and datasets. Fig. 5 visualizes the performance of GenNAS 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 [30], RSPS [23], DARTS-V1 [3], DARTS-V2, GDAS [58], SETN [59], and ENAS [60]. 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 [31] or groundtruth query to compensate the low rank correlations. We fairly use a simple yet effective regularized evolutionary algorithm [1] 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× faster. On NASBench-201, GenNAS finds better architectures than the state-of-the-art GDAS within 0.3 GPU hours, being 30× 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× 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× speed up. The regret score r at the
Table 4: Comparisons with state-of-the-art NAS methods on ImageNet under the mobile setting. * is the time for proxy task search.

<table>
<thead>
<tr>
<th>Method</th>
<th>Test Err. (%)</th>
<th>Params</th>
<th>FLOPS(M)</th>
<th>Search Cost (GPU days)</th>
<th>Search Method</th>
<th>Searched Method</th>
<th>Searched dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>top-1</td>
<td>top-5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NASNet-A [18]</td>
<td>26.0</td>
<td>8.4</td>
<td>5.3</td>
<td>564</td>
<td>2000</td>
<td>RL</td>
<td>CIFAR-10</td>
</tr>
<tr>
<td>AmoebaNet-C [1]</td>
<td>24.3</td>
<td>7.6</td>
<td>6.4</td>
<td>570</td>
<td>3150</td>
<td>evolution</td>
<td>CIFAR-10</td>
</tr>
<tr>
<td>PNAS [63]</td>
<td>25.8</td>
<td>8.1</td>
<td>5.1</td>
<td>588</td>
<td>225</td>
<td>SMBO</td>
<td>CIFAR-10</td>
</tr>
<tr>
<td>DARTS (2nd order) [1]</td>
<td>26.7</td>
<td>8.7</td>
<td>4.7</td>
<td>574</td>
<td>4.0</td>
<td>gradient-based</td>
<td>CIFAR-10</td>
</tr>
<tr>
<td>SNAS [64]</td>
<td>27.3</td>
<td>9.2</td>
<td>4.3</td>
<td>522</td>
<td>1.5</td>
<td>gradient-based</td>
<td>CIFAR-10</td>
</tr>
<tr>
<td>GDAS [33]</td>
<td>26.0</td>
<td>8.5</td>
<td>5.3</td>
<td>581</td>
<td>0.21</td>
<td>gradient-based</td>
<td>CIFAR-10</td>
</tr>
<tr>
<td>P-DARTS [63]</td>
<td>24.4</td>
<td>7.4</td>
<td>4.9</td>
<td>557</td>
<td>0.3</td>
<td>gradient-based</td>
<td>CIFAR-10</td>
</tr>
<tr>
<td>P-DARTS [32]</td>
<td>24.7</td>
<td>7.5</td>
<td>5.1</td>
<td>577</td>
<td>0.3</td>
<td>gradient-based</td>
<td>CIFAR-10</td>
</tr>
<tr>
<td>PC-DARTS [32]</td>
<td>25.1</td>
<td>7.8</td>
<td>5.3</td>
<td>586</td>
<td>0.1</td>
<td>gradient-based</td>
<td>CIFAR-10</td>
</tr>
<tr>
<td>TE-NAS [62]</td>
<td>26.2</td>
<td>8.3</td>
<td>6.3</td>
<td>-</td>
<td>0.05</td>
<td>training-free</td>
<td>CIFAR-10</td>
</tr>
<tr>
<td>PC-DARTS</td>
<td>24.2</td>
<td>7.3</td>
<td>5.3</td>
<td>597</td>
<td>3.8</td>
<td>gradient-based</td>
<td>ImageNet</td>
</tr>
<tr>
<td>TE-NAS</td>
<td>24.5</td>
<td>7.5</td>
<td>5.4</td>
<td>599</td>
<td>0.17</td>
<td>training-free</td>
<td>ImageNet</td>
</tr>
<tr>
<td>GenNAS-combo</td>
<td>25.1</td>
<td>7.8</td>
<td>4.8</td>
<td>559</td>
<td>0.04</td>
<td>evolution+few-shot</td>
<td>CIFAR-10</td>
</tr>
<tr>
<td>GenNAS-D14</td>
<td>24.3</td>
<td>7.2</td>
<td>5.3</td>
<td>599</td>
<td>0.7*+0.04</td>
<td>evolution+few-shot</td>
<td>CIFAR-10</td>
</tr>
</tbody>
</table>

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 [50] architecture.

On DARTS search space, we also perform the end-to-end search on ImageNet-1k [61] dataset. We fix the depth (layer) of the networks to be 14 and adjust the width (channel) so that the # of FLOPs is between 500M to 600M. We evaluate two strategies: one without task search using GenNAS-combo (see Table 1), and the other GenNAS-D14 with proxy task search on DARTS search space with depth 14 and initial channel 16. The training settings follow TENAS [62]. The results are shown in Table 4. We achieve top-1/5 test error of 25.1/7.8 using GenNAS-combo and top-1/5 test error of 24.3/7.2 using GenNAS-D14, which are on par with existing NAS architectures. GenNAS-combo is much faster than existing works, while GenNAS-D14 pays extra search time cost. Our next step is to investigate the searched tasks and demonstrate the generalization and transferrability of those searched tasks to further reduce the extra search time cost.

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 neural networks’ 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. With proxy search, GenNAS can deliver even better results. 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.

6 Acknowledgement

We thank IBM-Illinois Center for Cognitive Computing Systems Research (C3SR) for supporting this research. We thank all reviewers and the area chair for valuable discussions and feedback. This work utilizes resources [66] supported by the National Science Foundation’s Major Research Instrumentation program, grant #1725729, as well as the University of Illinois at Urbana-Champaign. P.L. is also partly supported by the 2021 JP Morgan Faculty Award and the National Science Foundation award HDR-2117997.
References


**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...

14
(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]