One-Shot Neural Architecture Search via Novelty Driven Sampling

Miao Zhang[1,2], **Huiqi Li**[1*], **Shirui Pan**[3], **Taoping Liu**[2] and **Steven Su**[2]
[1]School of Information and Electronics, Beijing Institute of Technology
[2]Faculty of Engineering and Information Technology, University of Technology Sydney
[3] Faculty of Information Technology, Monash University
{Miao.Zhang-2, Taoping.Liu}@student.uts.edu.au, huiqili@bit.edu.cn, shirui.pan@monash.edu, steven.su@uts.edu.au

Abstract

One-Shot Neural architecture search (NAS) has received wide attentions due to its computational efficiency. Most state-of-the-art One-Shot NAS methods use the validation accuracy based on inheriting weights from the supernet as the stepping stone to search for the best performing architecture, adopting a bilevel optimization pattern with assuming this validation accuracy approximates to the test accuracy after re-training. However, recent works have found that there is no positive correlation between the above validation accuracy and test accuracy for these One-Shot NAS methods, and this reward based sampling for supernet training also entails the rich-get-richer problem. To handle this deceptive problem, this paper presents a new approach, Efficient Novelty-driven Neural Architecture Search, to sample the most abnormal architecture to train the supernet. Specifically, a single-path supernet is adopted, and only the weights of a single architecture sampled by our novelty search are optimized in each step to reduce the memory demand greatly. Experiments demonstrate the effectiveness and efficiency of our novelty search based architecture sampling method.

1 Introduction

Neural architecture search (NAS) recently has attracted massive interests from deep learning community since it could relieve experts from a labor-intensive and time-consuming neural network design process [Elsken *et al.*, 2019; Liu *et al.*, 2018]. Despite its capacity to find competitive architectures, the computational complexity of NAS is highly expensive. Zoph *et al.* [2018] spends more than 1800 GPU days for reinforcement learning (RL) based NAS and Real *et al.* [2019] uses 450 GPUs for 7 days through evolutionary algorithm (EA) to train the model. Several works have been proposed to improve the efficiency of NAS, including performance prediction [Baker *et al.*, 2018], weight generation [Zhang *et al.*, 2019], and the popular weight sharing method [Pham *et al.*, 2018].

Weight sharing, also called One-Shot NAS [Pham *et al.*, 2018; Bender *et al.*, 2018], defines a supernet subsuming all possible architectures in the search space so that architectures can directly inherit weights from the supernet to avoid training from scratch. ENAS [Pham *et al.*, 2018] utilizes the validation accuracy with shared weights as the reward to optimize the RL based architecture sampling controller. Following up works [Liu *et al.*, 2019; Luo *et al.*, 2018] relax architectures into a continuous space, and optimize the architecture with respect to its validation accuracy through gradient descent. An important assumption in the weight-sharing NAS is that the validation accuracy with inherited weights from the supernet approximates to the test accuracy after re-training, or at least be highly predictive. However, several recent works [Bender *et al.*, 2018; Singh *et al.*, 2019; Sciuto *et al.*, 2020; Zhang *et al.*, 2020] point out that there is no positive correlation between the above validation accuracy and test accuracy for most One-Shot NAS methods. It indicates that we could not utilize the validation accuracy with inherited weights as useful feedback for controller improvement. In other words, searching for the optimal architecture based on weight sharing is deceptive because architectures with optimal performance on proxy tasks are not guaranteed to perform the best in the target task [Cai *et al.*, 2019].

Sciuto *et al.* [2020] and Singh *et al.* [2019] observed this deceptiveness that weight sharing is supposed to disorder the architecture rank, which usually deteriorates the performance of other architectures when training a new generated architecture. Benyahia *et al.* [Benyahia *et al.*, 2019] defined it as *multi-model forgetting* that the architecture learning in each step will deteriorate the performance of other architectures with shared connections, and make the proxy reward based on the supernet unreliable. Furthermore, since architectures with updated weights are supposed to have higher rewards, those performance reward based controllers have the potential to select those previously visited architectures with updated weights. Sampling architectures solely based on this deceptive reward without encouraging intelligent exploration entails the rich-get-richer problem [Li *et al.*, 2019] and leads to the local optima. As suggested by curiosity-driven exploration in deep reinforcement learning [Conti *et al.*, 2018], novelty-seeking could help the agent to learn new knowledge and avoid the local optima in RL domains with deceptive or sparse rewards. Different from the RL controller or gradient

*Corresponding Author

method, novelty search can alleviate this problem by encouraging the agent to visit unexplored areas rather than those areas with high performance. Compared with random sampling, novelty search further has the potential to fairly train the supernet in One-Shot NAS, as it always samples those untrained or less trained parts of the supernet to be trained, which could improve the predictive ability of the supernet [Chu *et al.*, 2019]. Instead of devising a complicated reward-based controller, we innovatively introduce novelty search to NAS, which samples architectures to train the supernet through novelty search. Since our method always samples architectures containing few shared connections with previously visited architectures, it could effectively relieve the multi-model forgetting that occurs during the supernet training and make the supernet more predictive. A weight-sharing based single-path model is adopted to reduce computational cost and memory storage, where all candidate architectures share weights and only the single-path weights are optimized in each step. Our contributions can be summarized as follows.

- Firstly, a novelty search based mechanism is innovatively applied to architecture sampling in One-Shot NAS for supernet training, where an efficient novelty-driven approach is devised to sample architectures without performance reward.
- Secondly, we adopt a weight-sharing based single-path model for neural architecture search, which could reduce not only the computational cost but also the memory storage significantly.
- Thirdly, extensive experimental results illustrate the superiority of our method, which achieves remarkable performance on benchmark datasets with efficiency. Our approach obtains the state-of-the-art test error of 2.51% for CIFAR-10 with only 7.5 hours of search time in a single GPU, and a competitive validation perplexity of 57.83 and a test perplexity of 55.88 on PTB with 4 hours search time. After transferring to larger datasets, our best models achieve a state-of-the-art test error of 16.56% on CIFAR-100, a competitive 26.66% on ImageNet, and a validation perplexity of 70.14 and a test perplexity of 69.31 on WT2. Our method also beats baselines on a NAS benchmark dataset.

2 Background

2.1 Neural Architecture Search

Neural architecture search (NAS) has attracted increasing attention to automatically design neural architectures to relieve human experts from the labor-intensive and time-consuming neural network design process. The search space of neural architecture $\mathcal{A}$ is generally represented as a directed acyclic graph (DAG), and the subgraph in the search space is denoted as $\alpha \in \mathcal{A}$ corresponding to a neural architecture $\mathcal{U}(\alpha, w)$ with weights w. NAS aims to find a subgraph α with the best validation loss after being trained on the training set, as:

$$\alpha^* = \underset{\alpha \in \mathcal{A}}{\texttt{argmin}}\ \mathcal{L}_{\texttt{val}}(\mathcal{U}(\alpha, w_\alpha)) \tag{1}$$

where $\mathcal{L}_{\texttt{val}}$ is the loss function on the validation set, and w_α are the weights of the architecture after being trained on the training set to minimize the training loss $\mathcal{L}_{train}$:

$$w_\alpha = \underset{w}{\texttt{argmin}}\ \mathcal{L}_{\texttt{train}}(\mathcal{U}(\alpha, w)) \tag{2}$$

Early NAS works adopt a nested manner to optimize weights and architectures, which samples numerous architectures to be trained on the training set and utilize EA or RL [Real *et al.*, 2019] to find promising architectures based on those evaluated architectures. These approaches have a high computational demand because evaluating an architecture is computationally expensive. A lot of NAS approaches are motivated by reducing computational cost, and a weight sharing mechanism (also called as One-Shot) is proposed [Pham *et al.*, 2018; Bender *et al.*, 2018], which dramatically reduces the search time to less than 1 GPU day. Instead of separate training architectures, weight sharing strategy encodes the whole search space $\mathcal{A}$ as a supernet $\mathcal{U}(\mathcal{A}, \mathcal{W})$, and all candidate architectures $\mathcal{U}(\alpha, w)$ directly inherit weights from the weights $\mathcal{W}$ of the supernet. Since only the supernet is trained in the architecture search phase, weight sharing NAS approaches can reduce the time for architecture search significantly. Weight sharing NAS contains two sequential steps:

1) supernet training:

$$\mathcal{W}_\mathcal{A} = \underset{\mathcal{W}}{\texttt{argmin}}\ \mathcal{L}_{\texttt{train}}(\mathcal{U}(\mathcal{A}, \mathcal{W})) \tag{3}$$

and 2) architecture selection:

$$\alpha^* = \underset{\alpha \in \mathcal{A}}{\texttt{argmin}}\ \mathcal{L}_{\texttt{val}}(\mathcal{U}(\alpha, \mathcal{W}_\mathcal{A}(\alpha))) \tag{4}$$

Recent works further relax architectures into a continuous space [Liu *et al.*, 2019; Dong and Yang, 2019; Xie *et al.*, 2019; Zhou *et al.*, 2019; Luo *et al.*, 2018], and alternatively optimize the supernet weights and architecture parameters based on bilevel optimization. Different from searching in the continuous space, Casale *et al.* [2019] propose a probabilistic approach PARSEC to sample architectures without continuous relaxation, where it uses an Importance-Weighted Monte Carlo empirical Bayes to define the architecture distribution.

Extensive experimental analysis in recent works [Bender *et al.*, 2018] shows it is possible to efficiently sample architectures for supernet training without any complex controllers for NAS. Guo *et al.* [2019] and Li *et al.* [2019] utilized the random sampling method to sample architectures for supernet training. The weight sharing is adopted in both of them to reduce the computational cost, and the memory requirements are the same as training a single architecture as only one path is activated in each step of the architecture search phase.

2.2 Novelty Search

Novelty search comes from the evolutionary community [Lehman and Stanley, 2011; Real *et al.*, 2019], which encourages the population to search for notably different areas to enhance the exploration. This approach utilizes the novelty as the stepping stone instead of the reward function, which makes it easy to avoid local optima in return. Previous novelty search based evolutionary algorithms [Lehman and Stanley, 2011] have shown their superiority in searching for small neural networks. Recent works on deep reinforcement learning [Conti *et al.*, 2018] also suggested that hybridized with

Algorithm 1 EN^{2}AS

Input: Training datase $\mathbb{D}_{train}$, validation dataset $\mathbb{D}_{val}$, test dataset $\mathbb{D}_{test}$, randomly initialized $\mathcal{W}$, initial architecture archive $A = \emptyset$, maximum number of stored architectures S, batch size b, training iteration T

for $i = 1, 2, ..., (T * \texttt{size}(\mathbb{D}_{train})/b)$ **do**
  if $\texttt{size}(A) < S$ **then**
    randomly sample an architecture α, update $\mathcal{W}_A(\alpha)$ by descending $\nabla_{\mathcal{W}_A(\alpha)}\mathcal{L}_{train}(\mathcal{W}_A(\alpha))$, and add architecture α into A;
  else
    randomly select α_θ^m from A, update it according Eq.(7), and replace α_θ^m with $\alpha_\theta^{m'}$;
    Apply argmax operation on the updated architecture to obtain α, and update the shared weights $\mathcal{W}_A(\alpha)$ by descending $\nabla_{\mathcal{W}_A(\alpha)}\mathcal{L}_{train}(\mathcal{W}_A(\alpha))$;
  end if
end for
Perform random search or EA on the trained supernet with validation dataset $\mathbb{D}_{val}$ to get α^* based on Eq.(8).
Retrain α^* and get the best performance on the test dataset.

Return: architecture α^* with best performance.

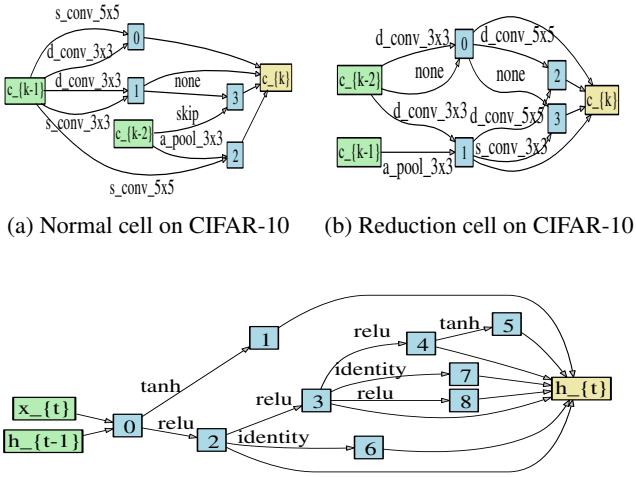
novelty search evolutionary algorithm could effectively avoid local optima in RL domains with deceptive reward functions. We investigate the effects of novelty search on neural architecture search in this paper and present how to use the novelty search mechanism as the controller to sample architectures for supernet training in the following section.

3 Methodology

In this section, we will describe our **E**fficient **N**ovelty-driven **N**eural **A**rchitecture **S**earch (EN^{2}AS). Algorithm 1 presents a simple implementation of EN^{2}AS, and we detailedly describe the architecture sampling for supernet training based on novelty search and also discuss architecture selection from trained supernet in following subsections.

3.1 Single Path Supernet Training based on Novelty Search

As described in Eq.(3), the inherited weights $\mathcal{W}_\mathcal{A}(\alpha)$ of architecture α from the supernet $\mathcal{A}$ should approximate to the optimal weights w_α or be highly predictive. Therefore, the key to weight sharing based NAS is how to train the supernet. As discussed in [Sciuto *et al.*, 2020; Bender *et al.*, 2018], a reward gradient-based architecture sampling controller is easy to be trapped in local optima, where there is no positive correlation between the validation accuracy with inherited weights and the test accuracy after re-training for such One-Shot NAS methods. Recent work [Conti *et al.*, 2018] on deep reinforcement learning demonstrates the effectiveness of novelty search as it could help the agent get out of local optimal when the reward function is very deceptive. In this paper, we utilize the novelty search to sample architectures for supernet training in One-Shot NAS.



(a) Normal cell on CIFAR-10 (b) Reduction cell on CIFAR-10

(c) Recurrent cell on PTB

Figure 1: Best cell structures found by our algorithm. For CNN cells, each node needs to select two former nodes with applied operations as its input. As to RNN cells, each node only selects one former node with applied operation as its input. The outputs for the three types of cells are the summation of outputs for all nodes.

The novelty search policy is defined as π and a behavior characterization $b(\pi)$ to describe its behavior. During the architecture search phase, every architecture α sampled from π is described as $b(\pi_\alpha)$ and added into archive A after calculating the novelty particular policy $N(b(\pi_\alpha), A)$. A simple and common novelty measurement is to calculate the mean distance of α and its k-nearest neighbors from A:

$$N(\alpha, A) = N(b(\pi_\alpha), A) = \frac{1}{|S|}\sum_{j \in S} \|b(\pi_\alpha) - b(\pi_j)\|_2$$
$$S = kNN(b(\pi_\alpha), A) = \{b(\pi_1), b(\pi_2), ..., b(\pi_k)\} \quad (5)$$

However, the distance calculation between neural architectures is not efficient because we need to compare all nodes and connections of two subgraphs, and calculating distances between the sampled architecture and all previously visited architectures in every search step. In this section, we introduce an archive based novelty search to relieve the high computational complexity for the novelty calculation. Given an archive A_θ containing a fixed number of continuous representation of sampled architectures as $\alpha_\theta^i = \alpha_\theta + \sigma\epsilon_i$, the gradient of expected novelty could be approximated as:

$$\nabla_{\alpha_\theta}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}[N(\alpha_\theta + \sigma\epsilon, A)|A] \approx \frac{1}{n\sigma}\sum_{i=1}^{n} N(\alpha_\theta^i, A)\epsilon_i \quad (6)$$

where $\epsilon_i \sim \mathcal{N}(0, I)$, α_θ^i is the *i-th* architecture with continuous parameters representation in the archive, n is the number of sampled perturbations to α_θ^t, and the archive is fixed at the beginning of the iteration and updated at the end. Eq. (6) demonstrates how to change the current architectures to increase the novelty of the archive, and we could update *m-th*

Method	Test Error (%) CIFAR-10	CIFAR-100	ImageNet	Param. (M)	Search Cost (GPU Days)	Memory Consumption	Supernet Optimization
NAO-WS [Luo *et al.*, 2018]	3.53	-	-	2.5	0.3	single path	gradient
ENAS [Pham *et al.*, 2018]	2.89	18.91	-	4.6	-	single path	RL
SNAS [Xie *et al.*, 2019]	2.85±0.02	20.09	27.3	2.8	1.5	whole supernet	gradient
BayesNAS [Zhou *et al.*, 2019]	2.81±0.04	-	26.5	3.40	**0.2**	whole supernet	gradient
MdeNAS [Zheng *et al.*, 2019]	2.51	-	-	4.06	0.16	single path	MDL
MdeNAS* [Zheng *et al.*, 2019]	2.87	17.61	26.8	3.78	0.16	single path	MDL
GDAS [Dong and Yang, 2019]	2.93	18.38	27.5	3.4	**0.21**	single path	gradient
DARTS (1st) [Liu *et al.*, 2019]	2.94	-	-	2.9	1.5	whole supernet	gradient
DARTS (2nd) [Liu *et al.*, 2019]	2.76±0.09	17.54	26.9	3.4	4	whole supernet	gradient
Random Search WS [2019]	2.85±0.08	17.63	-	4.3	2.7	single path	random
EN^{2}AS	**2.61±0.06**	**16.45**	**26.66**	3.1	**0.3**	single path	novelty search
EN^{2}AS with more epochs	**2.51±0.05**	-	-	3.1	**0.3**	single path	novelty search

Table 1: Comparison results with state-of-the-art weight sharing NAS methods on CIFAR-10, CIFAR-100 and ImageNet. "MdeNAS*" indicates that we reproduce the results based on the best-reported model in MdeNAS. All models are trained with 600 (250 for ImageNet) epochs, where the batch size is 96, and the initial channel is 36, to obtain the test error. We also further train our best architecture with 1000 epochs on CIFAR-10 and 500 epochs on ImageNet to achieve state-of-the-art results.

architecture in the archive according to:

$$\alpha_\theta^{m'} \leftarrow \alpha_\theta^m + \gamma \frac{1}{n\sigma} \sum_{i=1}^{n} N(\alpha_\theta^{m,i}, A)\epsilon_i \tag{7}$$

where γ is the stepsize. Based on Eq.(7), we only need to calculate the distance of the sampled architecture and an archive with a fixed number of architectures in every search step. It is straightforward to randomly select an architecture from the archive, and update it accordingly to optimize the novelty. In our practical implementation, only the architectures stored in the archive are continuous, and they are also applied with the **argmax** operation before calculating the distance to the sampled architectures.

3.2 Model Selection

Since evaluating an architecture is very efficient based on the trained supernet, it is possible to utilize a heuristic approach to find the most promising architecture, where random search and evolutionary algorithms are the two most common methods [Li and Talwalkar, 2019; Guo *et al.*, 2019]. In this paper, we adopt the validation accuracy as the optimizing goal in model selection as:

$$\underset{\alpha}{\texttt{maximize}} \quad ACC(\mathcal{W}_{\mathcal{A}}(\alpha)) \tag{8}$$

where $ACC(\mathcal{W}_{\mathcal{A}}(\alpha))$ is the validation accuracy of α with inhered weights from the supernet, and a baseline evolutionary algorithm is adopted to find the most promising architecture from the trained supernet.

4 Experiments and Results

The experimental design is following [Li and Talwalkar, 2019; Liu *et al.*, 2019; Xie *et al.*, 2019] for a fair comparison, which contains three stages: architecture search, architecture evaluation, and transfer to larger datasets. We perform our EN^{2}AS on small datasets, CIFAR-10 and PTB, to search for cell architectures on a smaller supernet architecture with fewer cells in the architecture search phase, and stack more multiple cells to construct larger architecture for full training and evaluation. Finally, the best-learned cells are also transferred to CIFAR-100, ImageNet and WT2 to investigate the transferability. We also evaluate the supernet predictive ability of our novelty based sampling method compared with two baselines in the following subsections [1].

4.1 Architecture Search for Convolutional Cells

Results on CIFAR-10

The comparison results on CIFAR-10 with the state-of-the-art NAS methods are demonstrated in Table 1. We report the results of the best found structure from 10 independent search experiments. It is impressive that the Random Search WS could obtain satisfactory results, which randomly sample architectures for supernet training. Random sampling strategy beats most reward-based sampling methods for One-Shot NAS with the same search space, except for DARTS (2nd) and BayesNet, which are with an elaborate controller. The result is also in line with the observation from [Bender *et al.*, 2018]. It is inspiring that the best architecture searched by our EN^{2}AS obtains the state-of-the-art test error on CIFAR-10 for weight sharing NAS. Our approach is also very efficient since the architecture search phase only costs about 7.5 hours (0.3 GPU day), and the memory consumption is the same as training a single architecture. The convolutional cell obtained by our EN^{2}AS is also very efficient, which has fewer parameters than most NAS methods.

Results on CIFAR-100 and ImageNet

The architecture evaluation setting on CIFAR-100 is the same as CIFAR-10, and the comparison results are also presented in Table 1. Our model could obtain a competitive result with 17.58% Top1 test errors with only 3.13M parameters. We further increase the number of initial filters from 36 to 50 (and the parameters increase to 5.88 M), and our network

[1] It is easy to reproduce our experimental results by replacing cell structures in DARTS [Liu *et al.*, 2019] with the structures shown in Fig.1. All codes, log files, and also trained models could be found in https://github.com/MiaoZhang0525/ENNAS_MASTER.

Method	Perplexity(PTB) Valid	Perplexity(PTB) Test	Perplexity(WT2) Valid	Perplexity(WT2) Test	Param. (M)	Search Cost (GPU Days)	Memory Consumption	Search Method
ENAS [Pham *et al.*, 2018]	60.8	58.6	72.4‡	70.4‡	24	0.5	single path	RL
DARTS (1st) [Liu *et al.*, 2019]	60.2	57.6	-	-	23	0.5	whole supernet	gradient
DARTS (2nd) [Liu *et al.*, 2019]	58.1	55.7	71.2	69.6	23	1	whole supernet	gradient
DARTS (2nd)* [Liu *et al.*, 2019]	**59.21**	**56.70**	-	-	23	1	whole supernet	gradient
GDAS [Dong and Yang, 2019]	59.8	57.5	71.0	69.4	23	0.4	single path	gradient
GDAS* [Dong and Yang, 2019]	60.23	57.69	-	-	23	0.4	single path	gradient
Random Search WS [2019]	57.8	55.5	-	-	23	0.25	single path	random
Random Search WS* [2019]	60.34	57.8	73.35	70.86	23	0.25	single path	random
EN^{2}AS	**59.28**	**57.26**	-	-	23	0.67	single path	novelty&reward
EN^{2}AS with 3600 epochs	**57.83**	**55.88**	**70.14**	**69.31**	23	0.67	single path	novelty&reward

Table 2: Comparison results with state-of-the-art NAS approaches on PTB and WT2. Since the results on PTB reported in these peer methods are with different training epochs, we reproduce the results of the best models reported in these approaches with the same experimental setting as ours, which are indicated by "*", for a fair comparison. ‡ means that the results are reproduced by DARTS with the same search space as ours. All models are trained with 1600 epochs with 64 batch size to obtain the perplexity, and we also further train our best-found architecture with 3600 epochs on to achieve competitive results.

achieves state-of-the-art results with a test error of 16.45% among all compared methods. The mobile setting on ImageNet also follows [Liu *et al.*, 2019] and we stacked the best found structure by 14 cells with batch size 128. Our model could obtain a competitive result with Top1/Top5 test errors as 26.66%/8.58% with only 4.5M parameters.

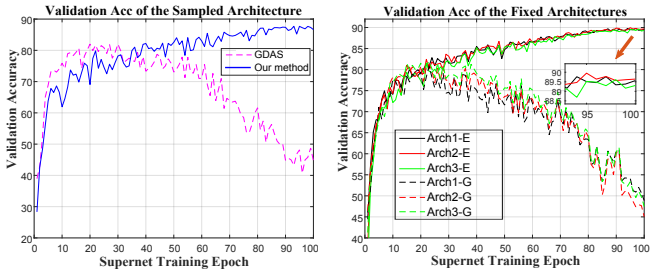
4.2 Architecture Search for Recurrent Cells

Results on PTB

The comparison results on PTB with the state-of-the-art manually-designed architectures and NAS methods are demonstrated in Table 2. We can find that the DARTS (2nd) achieves state-of-the-art results on PTB among those NAS methods, which obtains a validation perplexity of 59.21 and a test perplexity of 56.71 and shows the efficiency of gradient method in the recurrent search space. Our EN^{2}AS obtains a competitive validation perplexity of 59.28 and a test perplexity of 57.26, which is much better than DARTS (1st) and on par with the state-of-the-art NAS methods on PTB. As discussed before, our EN^{2}AS is a first-order iterative optimization based on novelty. The results clearly show that enhancing the exploration instead of sampling architecture based on performance reward could improve the supernet predictive ability, as evidenced by the fact that our EN^{2}AS beats the DARTS with first-order approximation. We further train our best found recurrent cell structure with more training epochs and achieve a competitive validation perplexity of 57.83 and test perplexity of 55.88.

Results on WT2

We also transfer those promising models obtained on PTB to WT2 following the experimental settings in [Liu *et al.*, 2019]. The embedding and hidden sizes are changed to 700, weight decay to 5×10^{-7}, hidden-node variational dropout to 0.15, and other hyperparameter settings are the same as PTB. The results of different models on WT2 are presented in Table 2. We train our best model with 3600 epochs on WT2 and achieve a state-of-the-art validation perplexity of 70.14 and a test perplexity of 69.31.

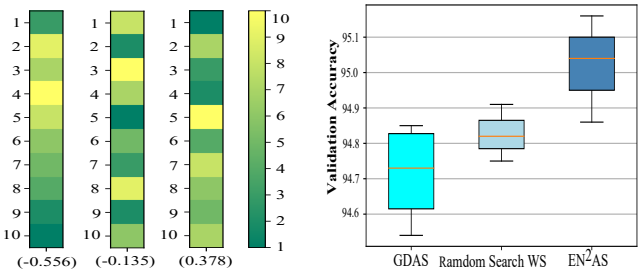


(a) Sampled Architecture (b) Fixed architectures

Figure 2: Validation accuracy of sampled architecture and fixed architectures during the supernet training for GDAS (dash lines) and EN^{2}AS (solid lines).

4.3 Empirical Comparison with Baselines

Supernet Training Comparison with Reward based Sampling Strategy. As discussed previously, the reward based controller entails the rich-get-richer problem [Li *et al.*, 2019], and the multi-model forgetting that occurs during the supernet training will also deteriorate the supernet's validation performance. In this section, we investigate the validation performance of architectures during the supernet training for our proposed novelty search based sampling strategy compared with reward based sampling strategy. We adopt the GDAS [Dong and Yang, 2019] as the reward based sampling baseline as it also only trains a single path in each step during the architecture search phase. We conduct this comparison experiment on CIFAR-10 and train the supernet with the two different sampling methods with 100 epochs, respectively. We tracked the validation accuracy of the sampled architecture in each step and also three fixed architectures through inheriting weights during the supernet training for the two sampling strategies. We present the validation accuracy of the sampled architectures during the supernet training in Fig.2 (a), and the validation accuracy of those fixed architectures in Fig.2 (b). It is straightforward that architectures are supposed to increase their validation accuracy with the supernet training. However,



(a) τ metric for GDAS, Random Search WS, and EN^{2}AS (b) Test accuracy

Figure 3: The τ metric and mean test accuracy for architectures obtained through different architecture sampling methods.

the performance of all those architectures shockingly gets worse during the supernet training for GDAS after several generations, as shown in Fig. 2, which demonstrates the existence of multi-model forgetting [Benyahia *et al.*, 2019] induced by reward-based sampling methods in One-Shot NAS that makes the supernet unreliable. Differently, our E^{2}NAS always samples abnormal architectures containing few shared connections with previously visited architectures to overcome this forgetting. Fig. 2 shows that the performance of architectures does not get worse during the supernet training based on novelty based sampling. It suggests that our proposed novelty search based sampling strategy can effectively relieve the multi-model forgetting and is more reliable than reward based sampling strategy.

Supernet Predictive Ability Evaluation. To demonstrate the effectiveness of our approach in relieving the rank disorder caused by weight sharing, we further conduct experiments to verify the supernet predictive ability of the proposed method compared with random sampling (Random Search WS, depicted as RS WS) and reward gradient based sampling (GDAS) in One-Shot NAS. We also replace the baseline evolutionary algorithm in the model selection of our EN^{2}AS with a random search for a fair comparison in this experiment. We conduct this comparison experiment on CIFAR-10 and also train the supernet with different sampling strategies for 100 epochs. Then we randomly sample 10 architectures and evaluate these architectures based on the three different trained supernets. We measure the correlation of architecture ranking based on weight sharing and retraining, and demonstrate the supernet predictive ability of three different sampling methods based on the Kendall Tau (τ) metric [Kendall, 1945]. Kendall Tau (τ) is to demonstrate the difference of ranking based on weight sharing and retraining for the three sampling methods. As shown in Figure 3 (a), the random sampling and gradient based sampling both obtain negative values that show the rank disorder in the two baselines. Our approach obtains a much better τ=0.378 with a positive correlation between the architecture ranking based on weight sharing and retraining, which indicates that the supernet trained based on novelty search sampling archives a better predictive ability. Since the supernet with better predictive ability tends to obtain better architectures, we further compare the retraining validation accuracy of sampled architectures from the trained supernet based on the three different architecture sampling methods. Figure 3 (b) plots the mean retraining validation accuracy, and we could observe that our novelty search based architecture sampling achieves the best results.

Method	Test Acc(%)	τ metric	**s-**τ metric
EN^{2}AS	**93.36±0.3**	**0.228±0.066**	**0.333**
RS WS [2019]	91.93±1.2	-0.016±0.100	0.111
GDAS [2019]	92.05±0.2	-0.067±0.109	-0.092

Table 3: Comparison with two baselines on NAS-Bench-102 dataset.

4.4 Experiments on Benchmark Dataset

The high computational cost of evaluating architectures is the major obstacle of analyzing One-Shot NAS methods, and several recent works try to build benchmark datasets [Ying *et al.*, 2019] to relieve this difficulty. We adopt NAS-Bench-102 [Dong and Yang, 2020] as a benchmark dataset to analyze our approach in this experiment. The search space in NAS-Bench-102 contains 4 nodes with 5 associated operations, which results in 15625 cell candidates. Although the search space in NAS-Bench-102 is much simpler than the common search space, the ground-truth test accuracy of all candidates in the search space is reported, which could greatly reduce the computational requirements in the analysis of One-Shot NAS methods. We run our EN^{2}AS on NAS-Bench-102 for three independent times with the same experimental settings in [Dong and Yang, 2020], and report the mean test accuracy of the best-found architectures in Table 3. To evaluate the supernet predictive ability, we further measure the Kendall Tau (τ) metric to demonstrate the difference of ranking based on supernet and ground truth for the three sampling methods. Apart from τ, we also calculate the **s-**τ to measure the stability of generated ranks from different runs, which is defined as $\frac{2}{N(N-1)}\sum_{1\leq i<j\leq N}\tau(R_i, R_j)$, where $N = 3$ in this experiment. We ranked 15 randomly generated architectures based on supernet and the ground-truth to obtain the (τ) and **s-**τ for the three methods, and the results are presented in Table 3, where our method beats the two baselines.

5 Conclusion and Future Work

This paper originally focuses on resolving the rich-get-richer problem in supernet training for weight-sharing neural architecture search, where a novelty search is proposed to enhance the exploration for architecture sampling during the supernet training. In particular, a novelty search mechanism is developed to efficiently find the most abnormal architecture, and the single-path model is adopted to greatly reduce computational and memory demand. Experimental results show the proposed approach could find the state-of-the-art or competitive CNN and RNN models, and also improve the predictive ability of the supernet in one-shot NAS. In our future work, we will focus on leveraging human knowledge in neural architecture search to enhance its transferable ability. Furthermore, how to use graph neural networks [Wu *et al.*, 2020] in NAS is also one of our future work directions.

References

[Baker *et al.*, 2018] Bowen Baker, Otkrist Gupta, Ramesh Raskar, and Nikhil Naik. Accelerating neural architecture search using performance prediction. *ICLR*, 2018.

[Bender *et al.*, 2018] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In *ICML*, pages 549–558, 2018.

[Benyahia *et al.*, 2019] Yassine Benyahia, Kaicheng Yu, Kamil Bennani Smires, Martin Jaggi, Anthony C Davison, Mathieu Salzmann, and Claudiu Musat. Overcoming multi-model forgetting. In *ICML*, pages 594–603, 2019.

[Cai *et al.*, 2019] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *ICLR*, 2019.

[Casale *et al.*, 2019] Francesco Paolo Casale, Jonathan Gordon, and Nicolo Fusi. Probabilistic neural architecture search. *arXiv preprint arXiv:1902.05116*, 2019.

[Chu *et al.*, 2019] 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.

[Conti *et al.*, 2018] Edoardo Conti, Vashisht Madhavan, Felipe Petroski Such, Joel Lehman, Kenneth Stanley, and Jeff Clune. Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. In *NeurIPS*, pages 5027–5038, 2018.

[Dong and Yang, 2019] Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. In *CVPR*. IEEE Computer Society, 2019.

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

[Elsken *et al.*, 2019] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *JMLR*, 20(55):1–21, 2019.

[Guo *et al.*, 2019] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling, 2019.

[Kendall, 1945] Maurice G Kendall. The treatment of ties in ranking problems. *Biometrika*, 33(3):239–251, 1945.

[Lehman and Stanley, 2011] Joel Lehman and Kenneth O Stanley. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation*, 19(2):189–223, 2011.

[Li and Talwalkar, 2019] Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. *arXiv preprint arXiv:1902.07638*, 2019.

[Li *et al.*, 2019] Xiang Li, Chen Lin, Chuming Li, Ming Sun, Wei Wu, Junjie Yan, and Wanli Ouyang. Improving one-shot nas by suppressing the posterior fading. *arXiv preprint arXiv:1910.02543*, 2019.

[Liu *et al.*, 2018] Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical representations for efficient architecture search. *ICLR*, 2018.

[Liu *et al.*, 2019] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *ICLR*, 2019.

[Luo *et al.*, 2018] Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu. Neural architecture optimization. In *NeurIPS*, pages 7816–7827, 2018.

[Pham *et al.*, 2018] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In *ICML*, pages 4092–4101, 2018.

[Real *et al.*, 2019] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. *AAAI*, 2019.

[Sciuto *et al.*, 2020] Christian Sciuto, Kaicheng Yu, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. Evaluating the search phase of neural architecture search. In *ICLR*, 2020.

[Singh *et al.*, 2019] Prabhant Singh, Tobias Jacobs, Sebastien Nicolas, and Mischa Schmidt. A study of the learning progress in neural architecture search techniques. *arXiv preprint arXiv:1906.07590*, 2019.

[Wu *et al.*, 2020] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *TNNLS*, 2020.

[Xie *et al.*, 2019] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: stochastic neural architecture search. *ICLR*, 2019.

[Ying *et al.*, 2019] Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. Nas-bench-101: Towards reproducible neural architecture search. In *ICML*, pages 7105–7114, 2019.

[Zhang *et al.*, 2019] Chris Zhang, Mengye Ren, and Raquel Urtasun. Graph hypernetworks for neural architecture search. *ICLR*, 2019.

[Zhang *et al.*, 2020] Miao Zhang, Huiqi Li, Shirui Pan, Xiaojun Chang, and Steven Su Su. Overcoming multi-model forgetting in one-shot nas with diversity maximization. In *CVPR*. IEEE Computer Society, 2020.

[Zheng *et al.*, 2019] Xiawu Zheng, Rongrong Ji, Lang Tang, Baochang Zhang, Jianzhuang Liu, and Qi Tian. Multinomial distribution learning for effective neural architecture search. In *ICCV*, 2019.

[Zhou *et al.*, 2019] Hongpeng Zhou, Minghao Yang, Jun Wang, and Wei Pan. Bayesnas: A bayesian approach for neural architecture search. In *ICML*, pages 7603–7613, 2019.

[Zoph *et al.*, 2018] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *CVPR*, pages 8697–8710, 2018.