One-Shot Neural Architecture Search via Novelty Driven Sampling

Miao Zhang1,2, Huiqi Li1*, Shirui Pan3, Taoping Liu2 and Steven Su2

1School of Information and Electronics, Beijing Institute of Technology
2Faculty of Engineering and Information Technology, University of Technology Sydney
3Faculty 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 [Elskens 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].

*Corresponding Author

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...
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 predictable. 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 $\alpha$ with the best validation loss after being trained on the training set to minimize the training loss $L_{\text{train}}$:

$$w_{\alpha} = \arg\min_w L_{\text{train}}(\mathcal{U}(\alpha, w))$$

(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 separately 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:

$$\mathcal{W}_\mathcal{A} = \arg\min_{\mathcal{W}} L_{\text{train}}(\mathcal{U}(\mathcal{A}, \mathcal{W}))$$

(3)

and 2) architecture selection:

$$\alpha^* = \arg\min_{\alpha \in \mathcal{A}} L_{\text{val}}(\mathcal{U}(\alpha, \mathcal{W}_\mathcal{A}(\alpha)))$$

(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^2AS

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

1: for \( i = 1, 2, \ldots, (T \times \text{size}(\mathcal{D}_{\text{train}}))/b \) do
2: if \( \text{size}(A) < S \) then
3: randomly sample an architecture \( \alpha \), update \( \mathcal{W}_A(\alpha) \) by descending \( \nabla \mathcal{W}_A(\alpha) \mathcal{L}_{\text{train}}(\mathcal{W}_A(\alpha)) \), and add architecture \( \alpha \) into \( A \);
4: else
5: randomly select \( \alpha^m_0 \) from \( A \), update it according Eq.(7), and replace \( \alpha^m_0 \) with \( \alpha^{m'}_0 \);
6: Apply argmax operation on the updated architecture to obtain \( \alpha \), and update the shared weights \( \mathcal{W}_A(\alpha) \) by descending \( \nabla \mathcal{W}_A(\alpha) \mathcal{L}_{\text{train}}(\mathcal{W}_A(\alpha)) \);
7: end if
8: end for
9: Perform random search or EA on the trained supernet with validation dataset \( \mathcal{D}_{\text{val}} \) to get \( \alpha^* \) based on Eq.(8).
10: Retrain \( \alpha^* \) and get the best performance on the test dataset.

Return: architecture \( \alpha^* \) with best performance.

The 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 Efficient Novelty-driven Neural Architecture Search (EN^2AS). Algorithm 1 presents a simple implementation of EN^2AS, 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 \( \mathcal{W}_A(\alpha) \) of architecture \( \alpha \) from the supernet \( A \) should approximate to the optimal weights \( w_{opt} \) 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.

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

\[
N(\alpha, A) = N(b(\pi, A), A) = \frac{1}{|S|} \sum_{\beta \in S} ||b(\pi, A) - b(\pi, \beta)||_2
\]

\[
S = kN N(b(\pi, A), A) = \{b(\pi_1), b(\pi_2), \ldots, b(\pi_k)\}
\]

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 \( A \) containing a fixed number of continuous representation of sampled architectures as \( \alpha^m = \alpha + \sigma \epsilon_i \), the gradient of expected novelty could be approximated as:

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

where \( \epsilon_i \sim \mathcal{N}(0, I) \), \( \alpha^m_i \) is the \( i \)-th architecture with continuous parameters representation in the archive, \( n \) is the number of sampled perturbations to \( \alpha^m_i \), 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
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:

\[
\text{maximize } \alpha \rightarrow ACC(W_A(\alpha)) \tag{8}
\]

where \(ACC(W_A(\alpha))\) is the validation accuracy of \(\alpha\) with inherited 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.

### 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.
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 based sampling strategy. The results clearly show that enhancing the exploration instead of sampling architecture based on novelty can improve the supernet predictive ability, as evidenced by the fact that our EN^2AS 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 $5 \times 10^{-7}$, 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.

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.

<table>
<thead>
<tr>
<th>Method</th>
<th>Perplexity(PTB)</th>
<th>Perplexity(WT2)</th>
<th>Param. (M)</th>
<th>Search Cost (GPU Days)</th>
<th>Memory Consumption</th>
<th>Search Method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Valid Test</td>
<td>Valid Test</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENAS [Pham et al., 2018]</td>
<td>60.8</td>
<td>58.6</td>
<td></td>
<td>24</td>
<td>0.5</td>
<td>single path RL</td>
</tr>
<tr>
<td>DARTS (1st) [Liu et al., 2019]</td>
<td>60.2</td>
<td>57.6</td>
<td></td>
<td>23</td>
<td>0.5</td>
<td>whole supernet gradient</td>
</tr>
<tr>
<td>DARTS (2nd) [Liu et al., 2019]</td>
<td>58.1</td>
<td>55.7</td>
<td></td>
<td>23</td>
<td>1</td>
<td>whole supernet gradient</td>
</tr>
<tr>
<td>DARTS (2nd)* [Liu et al., 2019]</td>
<td>59.21</td>
<td>56.70</td>
<td></td>
<td>23</td>
<td>1</td>
<td>whole supernet gradient</td>
</tr>
<tr>
<td>GDAS [Dong and Yang, 2019]</td>
<td>59.8</td>
<td>57.5</td>
<td></td>
<td>23</td>
<td>0.4</td>
<td>single path gradient</td>
</tr>
<tr>
<td>GDAS* [Dong and Yang, 2019]</td>
<td>60.23</td>
<td>57.69</td>
<td></td>
<td>23</td>
<td>0.4</td>
<td>single path gradient</td>
</tr>
<tr>
<td>Random Search WS [2019]</td>
<td>57.8</td>
<td>55.5</td>
<td></td>
<td>23</td>
<td>0.25</td>
<td>single path random</td>
</tr>
<tr>
<td>Random Search WS* [2019]</td>
<td>60.34</td>
<td>57.8</td>
<td></td>
<td>23</td>
<td>0.25</td>
<td>single path random</td>
</tr>
<tr>
<td>EN^2AS</td>
<td>59.28</td>
<td>57.26</td>
<td></td>
<td>23</td>
<td>0.67</td>
<td>single path novelty&amp;reward</td>
</tr>
<tr>
<td>EN^2AS with 3600 epochs</td>
<td>57.83</td>
<td>55.88</td>
<td></td>
<td>23</td>
<td>0.67</td>
<td>single path novelty&amp;reward</td>
</tr>
</tbody>
</table>

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,
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 EN\textsuperscript{2}AS 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 baseline 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\textsuperscript{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.

### 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\textsuperscript{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 ranked rules from different runs, which is defined as

\[
\sum_{1 \leq i < j \leq N} \tau(R_i, R_j), \tag{1}
\]

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


